



Lime  
Report

# **Руководство пользователя**

## СОДЕРЖАНИЕ

Введение	4
1 Сборка	5
2 Использование	6
3 Дизайнер	7
3.1 Быстрый доступ	7
3.2 Панели инструментов	8
3.3.1 Главная панель	8
3.3.2 Панель редактирования шрифта выделенных объектов	9
3.3.3 Панель форматирования текста	9
3.3.4 Панель редактирования расположения текстов	10
3.3.5 Панель редактирования границ	10
3.3.6 Панель объектов отчета	11
4 Источники данных	12
4.1 Инструментальное окно “Data Browser”	12
4.2 Объявление переменных	13
4.3 Соединения с базой данных непосредственно из генератора отчетов	14
4.4 Создание наборов данных в отчете	16
4.5 Внешние источники данных	19
5 Элементы отчета	22
5.1 Бэнд	22
5.2 Текст	25
5.3 Штрих код	29
5.4 Диаграмма	30
5.5 Изображение	31
5.6 Фигура	32
5.7 Горизонтальный лайаут	33
5.8 Вертикальный лайаут	33
6 Описание функций	34
6.1 ReportEngine	34
6.2 IDataSourceManager	35
6.3 IDbCredentialsProvider	36

	3	
6.4	IDataSource	36
6.5	IDataSourceHolder	37
6.6	PreparedPages	37
6.7	PreviewReportWidget	37
6.8	PrintRange	38
6.9	ReportDesignWindowInterface	38
6.10	ItemGeomerty	39
6.11	WatermarkSettings	39
6.12	ReportError	40
6.13	ReportSettings	40
7	Построение отчетов	41
7.1	Hello, World	41
7.2	Объект текст	42
7.3	Использование бэндов	44
7.4	Бэнеды-данные	46
7.5	Отчет “Список клиентов”	46
7.6	Объект рисунок	50
7.7	Отчет с картинками	51
7.8	Отображение многострочного текста	55
7.9	Разрыв данных	57
7.10	Печать данных в виде таблицы	58
7.11	Отчет с двумя уровнями данных (master-detail)	60

## ВВЕДЕНИЕ

Генератор отчетов для **Qt LimeReport** - кросс-платформенная C++ библиотека, написанная с использованием Qt framework и предназначенная для разработчиков программного обеспечения, которые хотят добавить в свое Qt приложение возможность формирования отчетов или печатных форм, генерируемых на основании шаблона.

Дизайнер отчетов, входящий в состав этой библиотеки, позволит быстро и интуитивно-понятно сформировать шаблон печатной формы, который может быть сохранен в формате XML и использован в дальнейшем для генерации страниц отчета.

Полученные таким образом страницы могут быть направлены на предварительный просмотр, в PDF файл или на принтер. В качестве источника данных поддерживается SQL база данных или данные, переданные из приложения с использованием интерфейса QAbstractTableModel.

Также из приложения могут быть проинициализированы переменные, которые доступны в качестве параметров запросов к базе данных.

Задача LimeReport - оснастить Ваше приложение функционально-богатым и в тоже время простым в использовании инструментом генерации отчетов, которым смогут пользоваться даже неискушенные в информационных технологиях пользователи.

Основные возможности:

- Кросс-платформенность
- Встроенный дизайнер отчетов
- Встроенный предварительный просмотр
- Внешнее или внутреннее подключение к источнику данных
- Возможность передавать параметры для внутренних запросов к базе данных из внешнего приложения
- Различные типы бэндов позволяющие создать отчет любой сложности.
- Верхний, нижний колонтитулы страницы.
- Группировка данных (GroupHeader, GroupFooter, Subdetail, SubdetailHeader, SubdetailFooter)
- Групповые функции (SUM, COUNT, AVG, MIN, MAX)
- Элементы отчета : Текст, Геометрические фигуры(линия, эллипс, прямоугольник), Изображение
- Возможность объединять элементы в горизонтальную группу
- Возможность использования HTML для форматирования полей вывода
- Возможность использования скриптов в для формирования выходных данных
- Автоматическое вычисление высоты бэнда
- И многое другое ...

## 1 СБОРКА

Для сборки LimeReport с помощью Qt, необходимо:

1. скачать последнюю версию LimeReport с репозитория на ресурсе <https://github.com/fralx/LimeReport>;
2. распаковать LimeReport в папку, в пути которой не содержатся символы кириллицы и пробелов, дальнейшем будем ссылаться на данную папку, используя псевдоним `<LimeReportSrc>`;
3. открыть в программе Qt Creator файл `limereport.pro`, расположенный в папке `<LimeReportSrc>`;
4. на панели слева выбрать пункт «Проекты»;
5. убрать галочку с пункта «Теневая сборка»;
6. запустить процесс сборки, вызвав пункт меню «Сборка → Собрать проект limereport»;
7. в случае отсутствия ошибок на предыдущем шаге в папке `<LimeReportSrc>` появляется папка `build`, содержащая сборку LimeReport под текущую платформу.

Для сборки LimeReport с помощью терминала в ОС Linux, необходимо:

1. скачать LimeReport 1.5.0 с репозитория на ресурсе <https://github.com/fralx/LimeReport>;
2. распаковать LimeReport 1.5.0 в папку, в пути которой не содержатся символы кириллицы и пробелов, дальнейшем будем ссылаться на данную папку, используя псевдоним `<LimeReportSrc>`;
3. открыть терминал и при помощи команды `cd` перейти в папку `<LimeReportSrc>`;
4. выполнить команду: `qmake limereport.pro`;
5. в случае ошибки на предыдущем шаге, необходимо выполнить установку компилятора GCC командой: `sudo apt install build-essential`;
6. выполнить команду `make`. На данном шаге возможны аварийные завершения процесса сборки. Наиболее распространенные ошибки приведены в таблице 1.

Таблица 1 — Типовые ошибки сборки.

Ошибка	Описание	Способ решения
Unknow module(s) in qt: qml	Не найден модуль для Qt: QML	<code>sudo apt-get install qtdeclarative5-dev</code>
Unknow module(s) in qt: svg	Не найден модуль для Qt: SVG	<code>sudo apt-get install libqt5svg5*</code>
Unknow module(s) in qt: uitools	Не найден модуль для Qt: uitools	<code>sudo apt-get install qttools5-dev</code>
Unknow module(s) in qt: designercomponents-private	Не найден модуль для Qt: designercomponents-private	<code>sudo apt-get install qttools5-private-dev</code>

7. в случае отсутствия ошибок на этапе сборки ПС в папке `<LimeReportSrc>` появляется папка *build*, содержащая сборку *LimeReport* под текущую платформу;

8. для более комфортной работы желательно добавить путь к дизайнеру (файл *LRDesigner*) в переменную окружения *PATH*, а путь к библиотекам (папка *lib*) - к переменной окружения *LD\_LIBRARY\_PATH*. И файл, и папка находятся внутри папки *build*.

Примечание: конкретные пути расположения файла *LRDesigner* и папки *lib* зависят от используемой версии Qt, аппаратной архитектуры и типа сборки. Например, путь к файлу *LRDesigner* может иметь вид: `<LimeReportSrc>/build/5.5.1/linux64/release/designer`.

Данный путь означает, что сборка проводилась с использованием библиотеки Qt5 с номером версии «5.1», на 64-битной ОС и тип сборки - «выпуск».

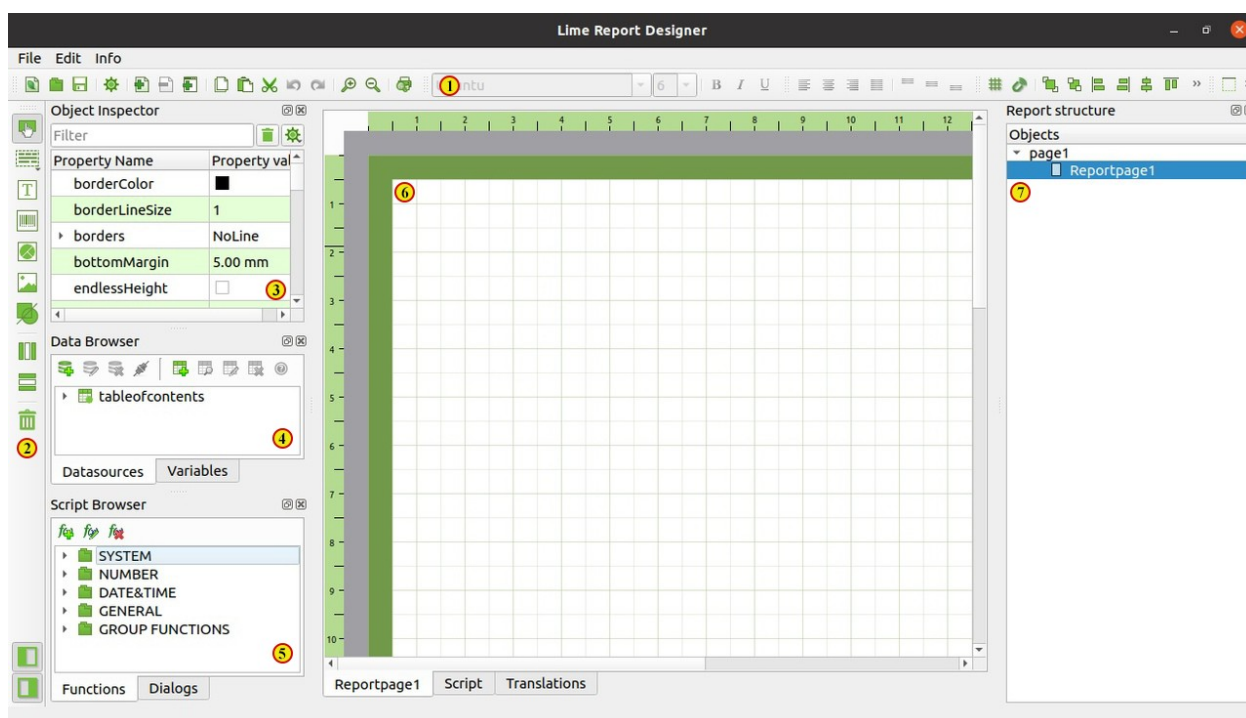
## 2 ИСПОЛЬЗОВАНИЕ

Для использования библиотеки *LimeReport* в разрабатываемом приложении, необходимо выполнить следующие шаги:

- выбрать пункт “Редактор”, расположенный в левой части окна программы QtCreator;
- нажать правую кнопку мыши, для которого необходимо подключить библиотеку *LimeReport*;
- в контекстном меню, нужно выбрать пункт «Добавить библиотеку»;
- тип библиотеки «Внешняя»;
- нажать кнопку «Далее»;
- выбрать необходимую платформу;
- выбрать файл библиотеки *liblimereport.so* для ОС Linux или *liblimereport.a* для ОС Windows, приблизительный путь `<LimeReportSrc>/build/5.5.1/linux64/release/lib`;
- указать путь к заголовочным файлам, приблизительный путь `<LimeReportSrc>/build/5.5.1/linux64/release/lib/include`;
- для ОС Windows необходимо убрать галочку на пункте «Добавить суффикс «d» для отладочной версии»;
- нажать кнопку «Далее»;
- нажать кнопку «Завершить»; повторить все шаги выше для файла *libQtZint.so* для ОС Linux или *libQtZint.a* для ОС Windows;
- в начало *pro*-файла проекта необходимо добавить строку: `QT += printsupport`.

После всех действий появляются необходимые записи в *pro*-файле проекта. Теперь можно использовать API функционал *LimeReport* для создания приложений поддерживающие создание шаблонных отчетов.

## 3 ДИЗАЙНЕР



1. Панель инструментов
2. Панель объектов отчета
3. Обзорщик объектов
4. Обзорщик данных
5. Обзорщик функций
6. Страница отчета
7. Обзорщик структуры отчета


### 3.1 БЫСТРЫЙ ДОСТУП

Ctrl+N	Новый отчет
Ctrl+O	Загрузить отчет
Ctrl+S	Сохранить
Ctrl+Shift+S	Сохранить как
Ctrl+P	Предварительный просмотр
Ctrl+Z	Отмена изменений
Ctrl+Shift+Z	Возврат отмененных изменений
Ctrl+C	Копировать
Ctrl+X	Вырезать
Ctrl+V	Вставить
Ctrl+Arrows	Переместить выделенный объект
Shift+Arrows	Изменить размер выделенных объектов
Del	Удалить выделенные объекты
Shift+Left mouse button	Создать область выделения

## 3.2 ПАНЕЛИ ИНСТРУМЕНТОВ

### 3.2.1 Главная панель



	Новый отчет
	Загрузить отчет
	Сохранить отчет
	Настройки
	Добавить страницу отчета
	Удалить страницу отчета
	Создание нового диалогового окна
	Копировать выделенное
	Вставить
	Вырезать
	Отмена изменений
	Возврат ранее отмененных изменений
	Увеличить масштаб просмотра страницы
	Уменьшить масштаб просмотра страницы
	Предварительный просмотр отчета



### 3.2.2 Панель редактирования шрифта выделенных объектов



	Шрифт
	Размер шрифта
	Жирность
	Курсив
	Подчеркивание













### 3.2.3 Панель форматирования текста



	Выравнивать текст по левому краю
	Располагать текст по центру
	Выравнивать текст по правому краю
	Выравнивать текст по ширине
	Прижимать текст к верхнему краю
	Располагать текст по центру
	Прижимать текст к нижнему краю






### 3.2.4 Панель редактирования расположения объектов



	Сетка
	Магнит
	Переместить выделенные объекты на передний план
	Переместить выделенные объекты на задний план
	Выровнять выделенные объекты по левому краю
	Выровнять выделенные объекты по правому краю
	Выровнять выделенные объекты вертикально по центру
	Выровнять выделенные объекты по верхнему краю
	Выровнять выделенные объекты по нижнему краю
	Выровнять выделенные объекты горизонтально по центру
	Сделать выделенные объекты одной высоты
	Сделать выделенные объекты одной ширины

### 3.2.5 Панель редактирования границ



	Обвести выделенные объекты сверху
	Обвести выделенные объекты снизу
	Обвести выделенные объекты слева
	Обвести выделенные объекты справа
	Снять границы

	Обвести объект целиком
---	------------------------

**3.2.6 Панель объектов отчета**



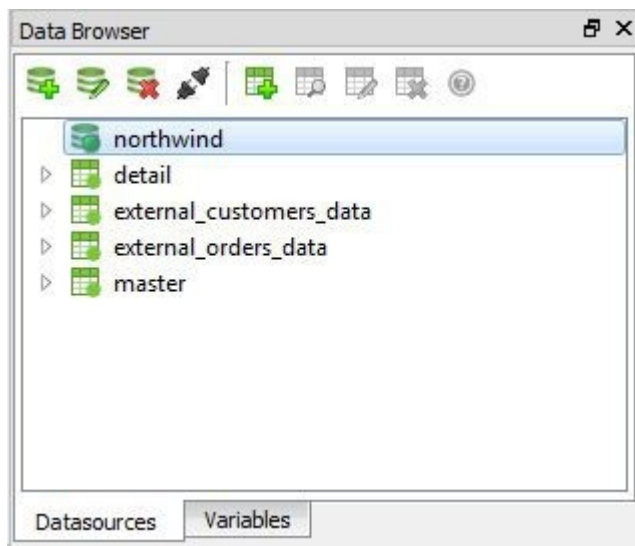
	Выбрать объект
	Вставить бэнд
	Вставить текстовую надпись
	Вставить штрих код (Требуется библиотека Zint)
	Вставить диаграмму
	Вставить картинку
	Вставить фигуру
	Объединить выделенные объекты в горизонтальный лайаут
	Объединить выделенные объекты в вертикальный лайаут
	Удалить объект

## 4 ИСТОЧНИКИ ДАННЫХ

В LimeReport предусмотрено несколько источников данных:

- Переменные, объявленные в отчете и доступные внешнему приложению.
- Наборы данных на основе SQL-запросов, использующие соединение с БД. Соединение с БД может быть инициализировано:
  - непосредственно из генератора отчетов
  - внешним приложением.
- Подключение внешних наборов данных через передачу генератору отчетов объекта, реализующего QAbstractItemModel.
- Реализация на стороне приложения методов передачи данных и подключение их к генератору отчетов посредством механизма SIGNAL-SLOT.

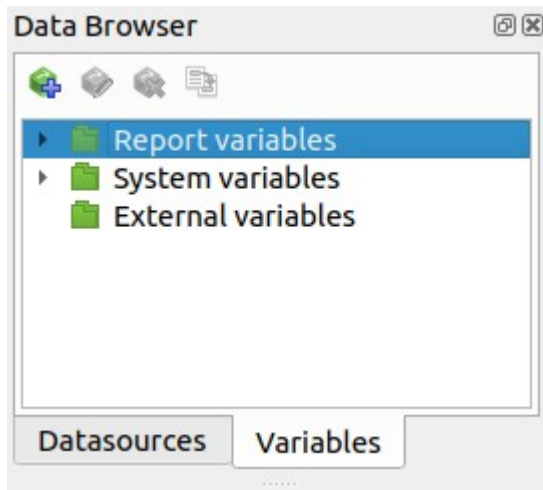
### 4.1 Инструментальное окно “Data Browser”




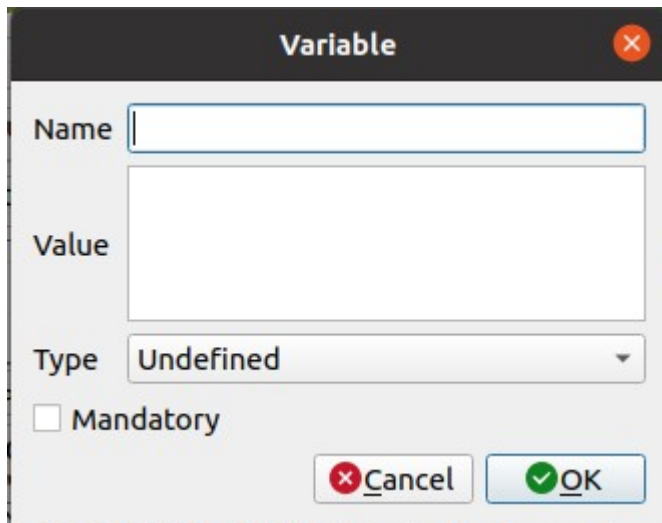
Данное окно предназначено для управления источниками данных. С его помощью задаются параметры соединений, переменных и наборов данных, которые впоследствии будут использованы в процессе построения отчета.



## 4.2 Объявление переменных

Параметры переменных можно установить на закладке “Variables” в инструментальном окне “Data Browser”.




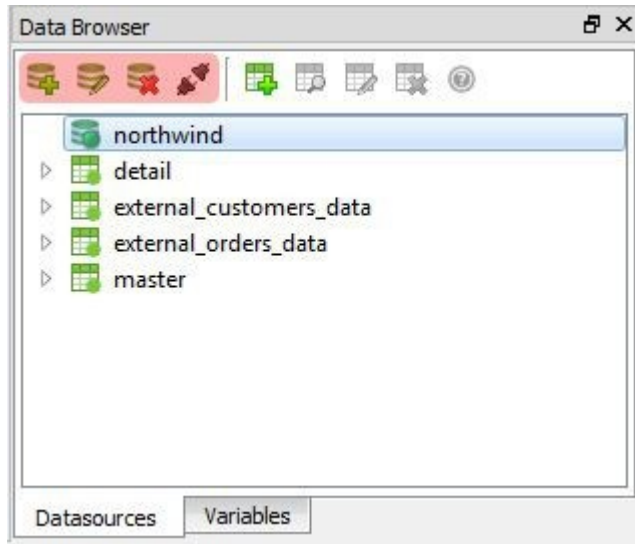
Объявление переменной производится кнопкой , далее посредством диалога “Variable” можно задать имя и значение переменной (значение переменной может быть изменено внешним приложением)



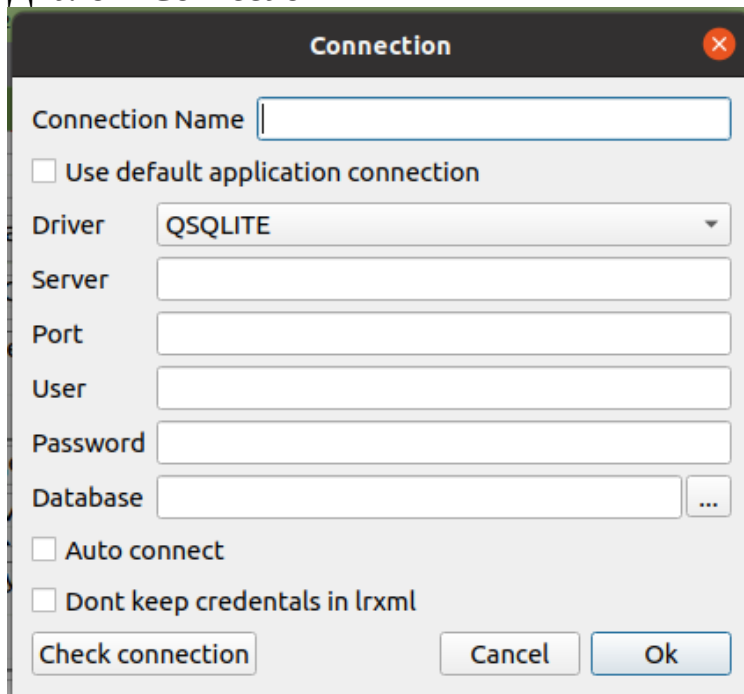
Используя кнопку  можно отредактировать объявление переменной. Кнопка  служит для удаления переменной.

### 4.3 Соединение с базой данных непосредственно из генератора отчетов

Параметры соединения с БД устанавливаются с помощью диалогового окна “Connection”, вызываемого посредством кнопки  (Add database connection), расположенной на панели инструментального окна “Data Browser”.



Диалог “Connection”



С помощью данного диалога настраиваются параметры соединения, такие как:


**Connection Name** – Название соединения для последующей идентификации в наборах данных, построенных на основе SQL запросов

**Driver** – Драйвер, посредством которого будет установлено соединения с базой данных Server - Адрес сервера, с которым будет установлено соединение

**User** – Имя пользователя



**Password** – Пароль

**Database** – Имя базы данных, к которой будет осуществляться подключение

**Auto connect** – Параметр, влияющий на то, будет ли автоматически устанавливаться соединение с БД непосредственно после загрузки отчета (создания описателя соединения). В случае, если параметр не отмечен, соединение будет устанавливаться в процессе генерации отчета или может быть принудительно активировано кнопкой  .

**Dont keep credentials in lrxml** – Параметр, влияющий на то, будет ли соединение связано с файлом отчета. В случае, если параметр отмечен файл отчета не будет связан с БД.

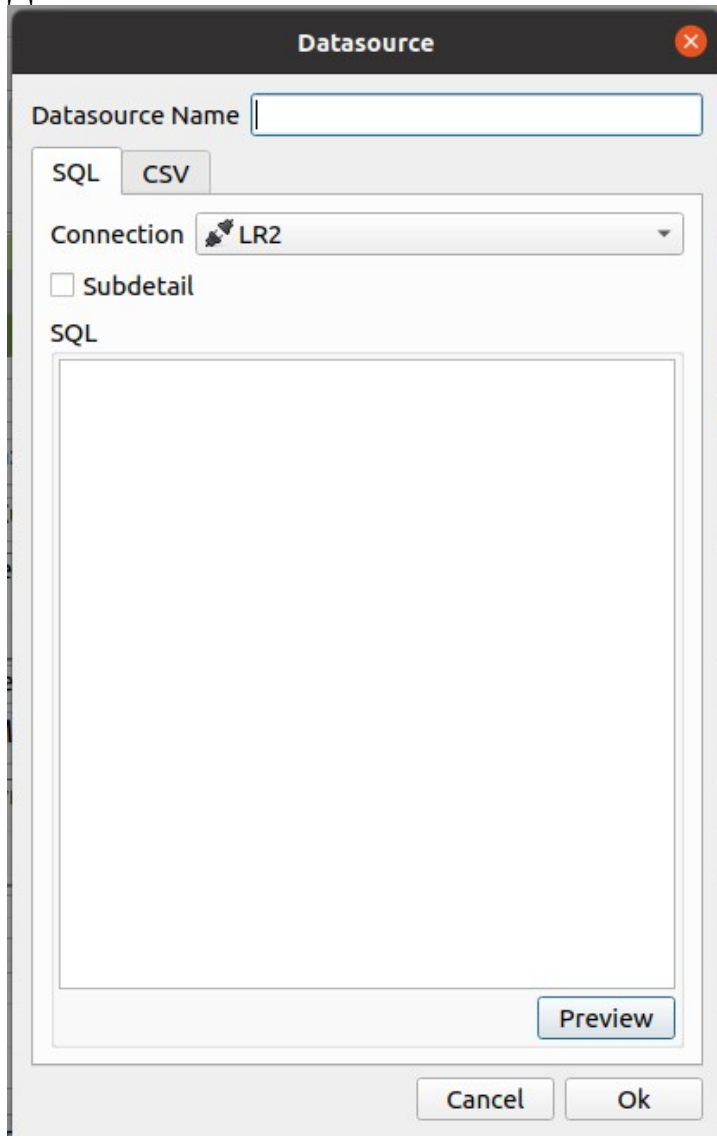
Кнопка «**Check connection**» позволяет проверить правильность заполнения параметров путем попытки установления соединения с указанными параметрами и вывода сообщения о результатах этой попытки.

После завершения конфигурации соединения параметры могут быть изменены с помощью кнопки  . Для удаления соединения служит кнопка  .

#### 4.4 Создание наборов данных в отчете

Параметры набора данных могут быть установлены с помощью диалогового окна «Datasource», вызываемого кнопкой , расположенной на панели инструментального окна «Data Browser».

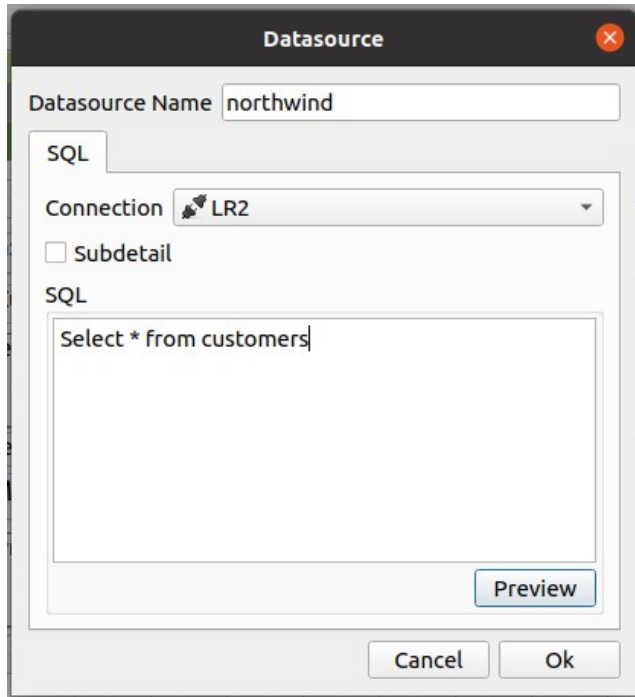
Диалог «Datasource»





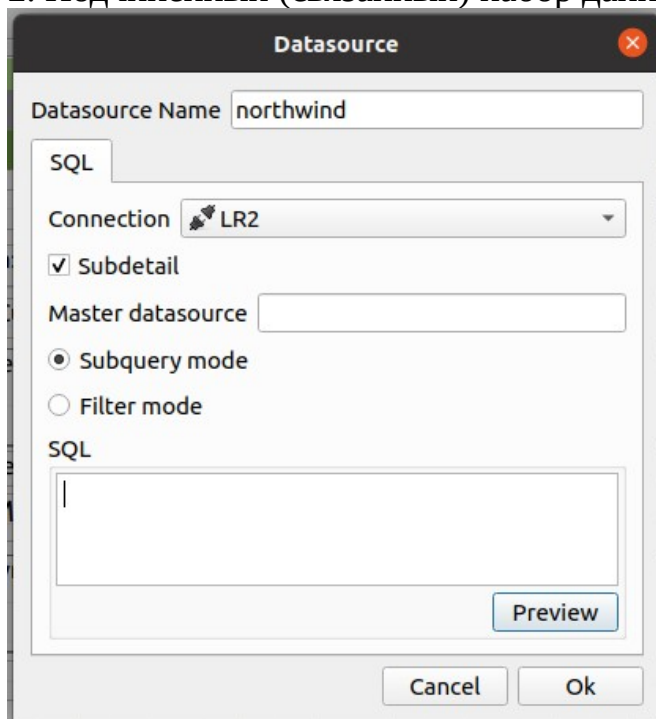
С помощью данного диалога возможно настроить несколько типов источников данных:

1. Источник данных, базирующийся на обычном SQL запросе к базе данных



Обычный запрос к базе данных может быть как независимым SQL запросом, так и запросом с параметрами, в качестве значений которых будут выступать переменные отчета. Обращение к переменным осуществляется посредством синтаксиса  $\$V\{\text{ИмяПеременной}\}$

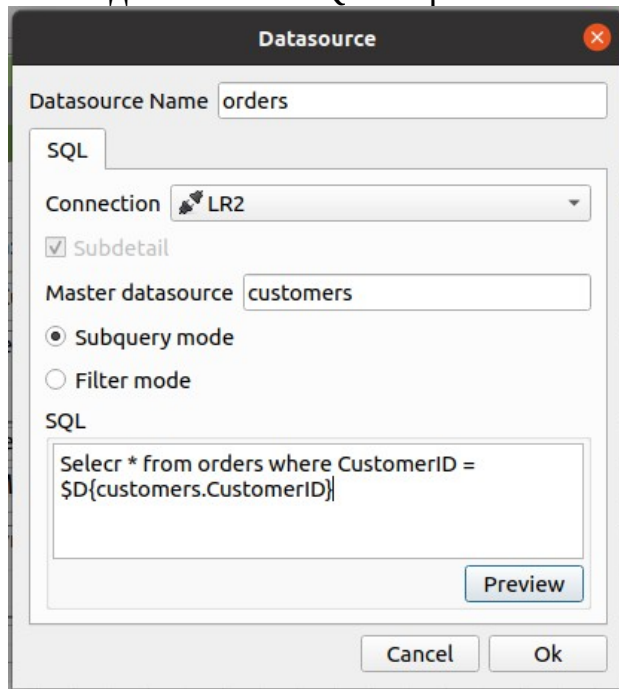
2. Подчиненный (связанный) набор данных



Подчиненный набор данных может использовать в качестве своих параметров другой источник данных (главный набор данных). Любой другой источник данных может выступать в качестве главного. Главный источник данных задается параметром “Master datasource”. Во время генерации отчета перемещение по главному набору данных сопровождается одновременным обновлением связанного набора. Для того, чтобы включить режим связанного набора, необходимо отметить чекбокс “Subdetail” в окне “Datasource”

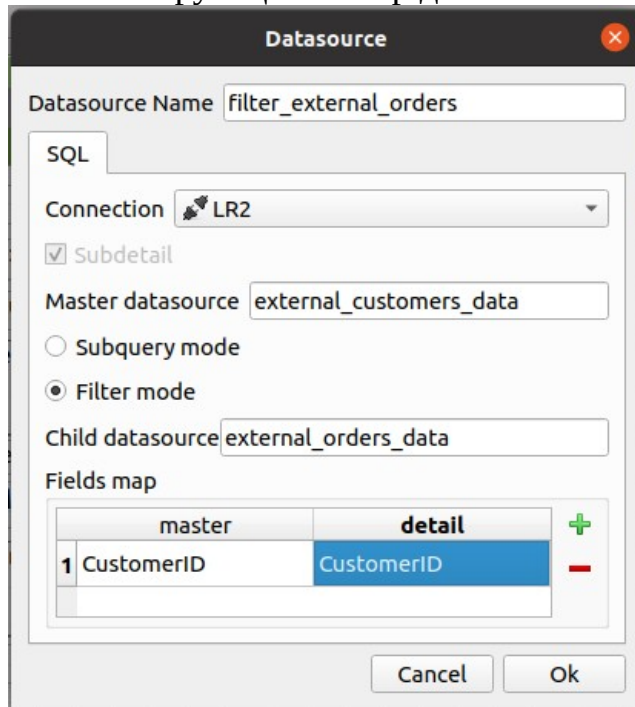
Подчиненные наборы могут быть двух видов:

### 2.1 Подчиненный SQL запрос



Для включения режима, подчиненного SQL запроса, необходимо выбрать «Sub query mode». Для доступа к полям главного набора данных используется синтаксис \$D{ИмяНабораДанных.ИмяПоля}.

## 2.1. Фильтрующий набор данных



Фильтрующий набор данных предназначен для случая, когда во время генерации отчета перемещение по главному набору данных должно приводить к фильтрации значений дочернего набора данных.

## 4.5 Внешние источники данных

Внешние источники данных могут быть двух видов:

1. Источник, реализующий интерфейс `QabstractDataModel`.  
Например, `QStringListModel`.

Подключение источника данного типа производится при помощи метода

```
Report->dataManager()->addModel(QString name, QAbstractDataModel* model, bool owned);
```

### Пример:

```
QStringList simpleData;
simpleData << "value1" << "value2" << "value3";
QStringListModel* stringListModel = new QStringListModel();
stringListModel->setStringList(simpleData);
report->dataManager()->addModel("string_list",stringListModel,true);
```

2. Источник, подключенный с помощью механизма SIGNAL-SLOT. Для использования данного метода подключения необходимо:

2.1. Реализовать метод (слот) получения данных с параметрами (`LimeReport::CallbackInfo info, QVariant &data`), где параметр `info` в

поле `dataType` (`info.dataType`) содержит информацию о том, какие данные должен вернуть этот метод через изменяемый параметр `data`:

- `LimeReport::CallbackInfo::IsEmpty` - `data (bool)` - есть ли данные в источнике
- `LimeReport::CallbackInfo::HasNext-data`(`bool`)-есть ли следующая строка в источнике
- `LimeReport::CallbackInfo::ColumnHeaderData` - `data(QString)` - наименование для колонки с индексом, указанным в `info.index` - это значение будет в дальнейшем использоваться для идентификации этой колонки в отчете и в методе получения значений
- `LimeReport::CallbackInfo::ColumnData` - `data(QVariant)` - значение для колонки с наименованием, указанным в `info.columnName` (это значение ранее было установлено через вызов метода с параметром `LimeReport::CallbackInfo::ColumnHeaderData`)

2.2. Реализовать метод (слот) перемещения по набору данных с параметрами (`const LimeReport::CallbackInfo::ChangePosType &type, bool &result`). Параметр `type` указывает, какого типа должно быть осуществлено перемещение, а изменяемый параметр `result` должен быть установлен в значение `true`, если перемещение было успешным, и `false` в обратном случае. Параметр `type` может принимать следующие значения:

- `First` - перейти к первой позиции набора данных
- `Next` - перейти к следующей позиции

2.3. Создать новый (`ICallbackDatasource`) источник данных с помощью метода `report->dataManager()->createCallbackDatasource()` и подключить ранее созданные методы к сигналам: `getCallbackData` - первый метод, `changePos` - второй.

### Пример:

```
class MainWindow : public QMainWindow
{
...
 QSqlQuery* m_customers;
.....
}
void MainWindow::prepareData(QSqlQuery* ds, LimeReport::CallbackInfo info, QVariant
&data)
{
```

```

switch (info.dataType) {
    case
        LimeReport::CallbackInfo::IsEmpty: data = !ds->first();
        break;
    case
        LimeReport::CallbackInfo::HasNext: data = ds->next();
        if (data.toBool()) ds->previous();
        break;
    case
        LimeReport::CallbackInfo::ColumnHeaderData:
        if (info.index < ds->record().count())
            data = ds->record().fieldName(info.index);
            break;
        case LimeReport::CallbackInfo::ColumnData: data = ds->value(info.columnName);
            break;
    }
}

void MainWindow::slotGetCallbackData(LimeReport::CallbackInfo info, QVariant &data)
{
    if (!m_customers) return; prepareData(m_customers, info,data);
}

void MainWindow::slotChangePos(const LimeReport::CallbackInfo::ChangePosType &type,
bool &result)
{
    QSqlQuery* ds = m_customers; if (!ds) return;

    if (type == LimeReport::CallbackInfo::First) result = ds->first();
    else result = ds->next();
    // В этом методе может быть реализовано обновление зависимого источника
    // данных m_orders->bindValue(":id",m_customers->value("CustomerID"));
    // m_orders->exec();
}
{
    ...
    LimeReport::ICallbackDatasource * callbackDatasource =
        report->dataManager()->createCallbackDatasouce();
    connect(
        callbackDatasource, SIGNAL(getCallbackData (LimeReport::CallbackInfo,Qvariant&)),
        this, SLOT(slotGetCallbackData(LimeReport::CallbackInfo,QVariant&)));
    connect(
        callbackDatasource,
        SIGNAL(changePos(constLimeReport::CallbackInfo::ChangePosType&,bool&)),
        this, SLOT
        (slotChangePos(constLimeReport::CallbackInfo::ChangePosType&,boo&)));
    report->dataManager()->addCallbackDatasource(callbackDatasource,"master");
    ...
}

```

## 5 ЭЛЕМЕНТЫ ОТЧЕТА

### 5.1 - БЭНД

Элемент “контейнер” предназначен для размещения других элементов отчета. Бэнд может быть нескольких типов:

- Report Header - заголовок отчета
- Report Footer - завершение отчета
- Page Header - верхний колонтитул страницы отчета
- Page Footer - нижний колонтитул страницы отчета
- Data - данные отчета
- Data Header - заголовок данных отчета
- Data Footer - завершение данных отчета
- SubDetail - подчиненные данные отчета
- SubDetailHeader - заголовок подчиненных данных
- SubDetailFooter - завершение подчиненных данных
- GroupHeader - заголовок группы
- GroupFooter - завершение группы
- Tear-off band - разрыв группы

#### Общие для всех бэндов свойства:

autoHeight	Автоматический подбор высоты
backgroundBrushStyle	Стиль кисти фона
backgroundColor	Цвет заливки бэнда
backgroundMode	TransparentMode – прозрачный режим OpaqueMode – непрозрачный режим
backgroundOpacity	Прозрачность бэнда
borderColor	Цвет границ
borderLineSize	Размер границ
borders	Границы
geometry	Размер и расположение объекта
geometryLocked	Блокирование размера и расположения объекта
keepBottomSpace	Сохранять отступ от нижней границы бэнда
keepTopSpace	Сохранять отступ от верхней границы бэнда
objectName	Имя объекта
printIfEmpty	Печатать, если на бэнде нет данных
splittable	Разделить бэнд, если он не влезает на страницу

**PageHeader, PageFooter**

printOnFirstPage	Печатать на первой странице
printOnLastPage	Печатать на последней странице

**ReportHeader**

printBeforePageHeader	Печатать перед заголовком страницы
-----------------------	------------------------------------

**ReportFooter**

maxScalePercent	Максимальный процент, на который можно уменьшить бэнд, если он не влезает на страницу. Если бэнд даже после сжатия не влезает на страницу, он будет перенесен полностью или частично, в зависимости от настроек бэнда
-----------------	---

**DataHeader**

columnsCount	Число столбцов
columnsFillDirection	Направление заполнения столбцов
datasource	Источник данных. DataHeader бэнд будет сформирован для каждой строки в источнике данных
keepFooterTogether	
keepSubdetailTogether	
sliceLastRow	Нарезать последний ряд
startFromNewPage	Начать с новой страницы
startNewPage	Начать с новой страницы, каждый раз
useAlternateBackgroundColor	Использовать альтернативный цвет фона
printAlways	Печатать, даже если DataHeader пустой
repeatOnEachRow	Повторяет в каждом ряду
repeatOnEachPage	Повторяет на каждой странице

**Data**

alternateBackgroundColor	Альтернативный цвет фона
columnsCount	Число столбцов
columnsFillDirection	Направление заполнения столбцов
keepFooterTogether	
keepSubdetailTogether	
sliceLastRow	Нарезать последний ряд
startFromNewPage	Начать с новой страницы
startNewPage	Начать с новой страницы, каждый раз
useAlternateBackgroundColor	Использовать альтернативный цвет фона
datasource	Источник данных. Data бэнд будет сформирован для каждой строки в источнике данных
keepFooterTogether	Если Report Footer не влезает на страницу, то он будет перенесен на следующую страницу совместно с последним экземпляром Data бэнда
sliceLastRow	Указывает генератору отчетов на то, можно ли разрезать последний экземпляр Data бэнда или его нужно перенести целиком

**DataFooter**

columnsCount	Число столбцов
columnsFillDirection	Направление заполнения столбцов
datasource	Источник данных. DataFooter бэнд будет сформирован для каждой строки в источнике данных
keepFooterTogether	
keepSubdetailTogether	
sliceLastRow	Нарезать последний ряд
startFromNewPage	Начать с новой страницы
startNewPage	Начать с новой страницы, каждый раз
useAlternateBackgroundColor	Использовать альтернативный цвет фона
printAlways	Печатать, даже если DataFooter пустой



**SubDetail**


datasource	Источник данных. SubDetail бэнд будет сформирован для каждой строки в источнике данных
columnsCount	Число столбцов
columnsFillDirection	Направление заполнения столбцов
keepFooterTogether	
alternateBackgroundColor	Альтернативный цвет фона
useAlternateBackgroundColor	Использовать альтернативный цвет фона

**SubDetailHeader, SubDetailFooter**

printAlways	Печатать, даже если SubDetail пустой
columnsCount	Число столбцов
columnsFillDirection	Направление заполнения столбцов

**GroupHeader**

groupFieldName	Поле, по которому осуществляется группировка. Экземпляр GroupHeader будет формироваться при смене значения в этом поле
columnsCount	Число столбцов
columnsFillDirection	Направление заполнения столбцов
keepGroupToogether	Печатать, даже если GroupHeader пустой
reprintOnEachPage	Перепечатывает на каждой странице
resetPageNumber	Сбрасывает номер страницы
splittable	
startNewPage	Начать с новой страницы, каждый раз

**5.2  - ТЕКСТ**

Элемент Текст служит для вывода надписей или содержимого полей источников данных.

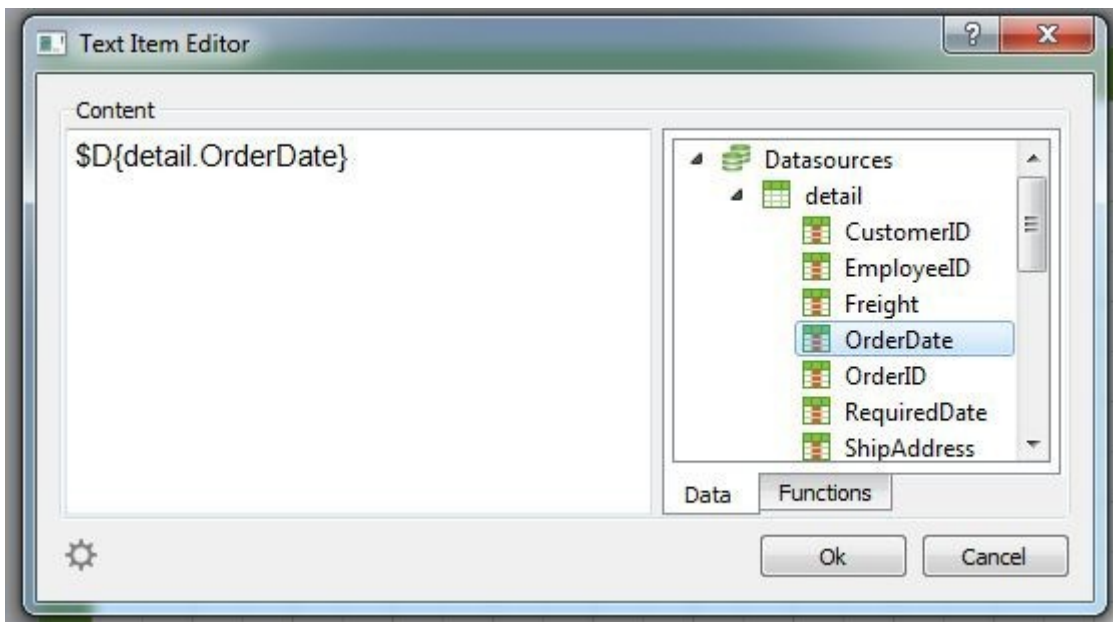
Параметры элемента Текст

adaptFontToSize	Адаптировать шрифт под размер элемента
alignment	Выравнивание текста по вертикали и горизонтали

allowHTML	Позволяет использовать HTML
allowHTMLInFields	Позволяет использовать HTML в полях
angle	Поворот надписи
autoHeight	Автоматический подбор высоты
autoWidth	Автоматический подбор ширины
backgroundBrushStyle	Стиль кисти фона
backgroundColor	Цвет заливки
backgroundMode	Режим заливки
backgroundOpacity	Прозрачность заливки
borderColor	Цвет границ
borderLineSize	Размер границ
borders	Границы
content	Содержимое объекта текст
fillInSecondPass	
FollowTo	
font	Шрифт
fontColor	Цвет шрифта
fontLetterSpacing	Интервал между буквами
foregroundOpacity	Прозрачность шрифта
geometry	Размер и расположение объекта
geometryLocked	Блокирование размера и расположения объекта
hideIfEmpty	Скрыть, если объект пустой
itemAlign	Выравнивание элемента
itemLocation	Расположение объекта (Страница или Бэнд)
lineSpacing	Межстрочный интервал
margin	Отступы
objectName	Имя объекта
replaceCRwithBR	
stretchToMaxHeight	Устанавливать значение высоты самого высокого объекта на бэнде
textIndent	Отступ текста

textLayoutDirection	Расположение текста в объекте
trimValue	Обрезать пробелы в начале и в конце надписи.
underlineLineSize	Размер линии подчеркивания
underLines	Подчеркивание
valueType	Тип текста
watermark	Устанавливает водяной знак

Для редактирования параметра «content» используется «Text Item Editor» вызов, которого осуществляется двойным кликом на элементе Текст.



«Content» может содержать: Текст, Значения переменных, Значения полей из набора данных, а также значения, формируемые посредством исполнения скрипта. Для вывода значений переменных используется синтаксис  $\$V\{\text{имяПеременной}\}$ , для создания переменной см. пункт «Объявление переменных», для вывода значения поля данных  $\$D\{\text{имяНабораДанных.ИмяПеременной}\}$ , для обращения к наборам данных см. пункт «Создание наборов данных в отчете», для исполнения скрипта  $\$S\{\text{телоСкрипта}\}$ .

В теле  $\$S\{\}$  можно использовать язык программирования JavaScript. Существуют также функции, которые можно использовать в  $\$S\{\}$ .

Категория **SYSTEM**:

**line**(BandName) - Выводит количество записей BandName.

Пример: в бэнде GroupBandHeader создаем текстовый объект в нем пишем  $\$S\{\text{line}(\text{«DataBand1»})\}$ . В предпросмотре отчета это текстовое поле будет содержать количество записей бэнда DataBand1.

Категория **NUMBER**:

**numberFormat**(Value, Format, Precision, Locale) — Изменяет формат числа.

Пример: `$${numberFormat(123456789.234, "f", 3, "en-EN")}`

Вывод: 123,456,789.234

**currencyFormat**(Value, Local) - изменяет формат валюты.

Пример: `$${currencyFormat(123456789.234, "ru-RU")}`

Вывод: 123 456 789.234 ₺

**currencyUSBasedFormat**(Value, CurrencySymbol) - изменяет формат валюты по американским стандартам

Категория **DATE&TIME**:

**dateFormat**(Value, Format, Locale) - изменяет формат даты.

Пример: `$${dateFormat(date(), "MM dd yyyy", "ru-RU")}`

Вывод: 06 21 2021

**timeFormat**(Value, Format) - изменяет формат времени.

Пример: `$${timeFormat(now(), "hh mm ss")}`

Вывод: 14 18 43

**dateTimeFormat**(Value, Format, Local) - изменяет формат времени и даты.

Пример: `$${dateTimeFormat(now(), "hh mm ss MM dd yyyy")}`

Вывод: 14 21 24 06 21 2021

**sectotimeFormat**(Value, Format) - возвращает время в формате Format для значения секунд Value.

Пример: `$${sectotimeFormat(123456, "hh mm ss")}`

Вывод: 34 17 36

**date()** - выводит системную дату.

**now()** - выводит системное время и дату.

Категория **GENERAL**:

**setVariable**(Name, Value) - установить переменной Name значение Value.

**getVariable**(Name) - возвращает значение переменной Name.

**getField**(Name) - возвращает значение поля Name.

**getFieldByKeyField**(Datasource, ValueField, KeyField, KeyFieldValue)

**getFieldByRowIndex**(FieldName, RowIndex)

**addBookmark**(UniqueIdentifier, Content)

**findPageIndexByBookmark**(UniqueIdentifier)

**addTableOfContentsItem**(Unique identifier, Content, Indent)

**clearTableOfContentsItem**()

**reopenDatasource**(datasourceName)

Категория **GROUP FUNCTION**:

**AVG** - агрегатная функция, которая выводит среднее значение по указанным полям или переменным.

Пример:  $\$S\{AVG("\$D(datasource.field)", "DataBand1")\}$

**COUNT** - агрегатная функция, которая выводит количество указанных полей или переменных.

Пример:  $\$S\{COUNT("\$D(datasource.field)", "DataBand1")\}$

**MAX** - агрегатная функция, которая выводит максимальное значение, среди указанных полей или переменных.

Пример:  $\$S\{MAX("\$D(datasource.field)", "DataBand1")\}$

**MIN** - агрегатная функция, которая выводит минимальное значение, среди указанных полей или переменных.

Пример:  $\$S\{MIN("\$D(datasource.field)", "DataBand1")\}$

**SUM** - агрегатная функция, которая выводит сумму указанных полей или переменных.

Пример:  $\$S\{SUM("\$D(datasource.field)", "DataBand1")\}$

### 5.3 - ШТРИХ КОД

Элемент для отображения штрих кодов

angle	Угол поворота штрих кода
backgroundColor	Цвет заливки
barcodeType	Тип штрих кода
barcodeWidth	параметр для QZint

borderColor	Цвет границ
borderLineSize	Размер границ
borders	Границы
content	Значение для отображения штрих кодом
datasource	Источник данных
field	Поле данных
foregroundColor	Цвет штрих кода
geometry	Размер и расположение объекта
geometryLocked	Блокирование размера и расположения объекта
hideIfEmpty	Скрыть, если объект пустой
hideText	Скрыть текст
inputMode	
itemAlign	Выравнивание элемента
itemLocation	Расположение объекта (Страница или Бэнд)
objectName	Имя объекта
pdf417CodeWords	параметр для QZint
securityLevel	параметр для QZint
stretchToMaxHeight	Устанавливать значение высоты самого высокого объекта на бэнде
testValue	Значение которое отображает штрих код в режиме разработки
whitespace	Свободное пространство

## 5.4 - ДИАГРАММА

Элемент для отображения диаграмм

borderColor	Цвет границ
borderLineSize	Размер границ
borders	Границы
chartTitle	Устанавливает заголовок диаграммы
chartType	Устанавливает тип диаграммы
datasource	Источник данных

drawLegendBorder	Рисует границу легенды диаграммы
geometry	Размер и расположение объекта
geometryLocked	Блокирование размера и расположения объекта
itemAlign	Выравнивание элемента
itemLocation	Расположение объекта (Страница или Бэнд)
labelsField	
legendAlign	Выравнивание легенды диаграммы
objectName	Имя объекта
series	
stretchToMaxHeight	Устанавливать значение высоты самого высокого объекта на бэнде
titleAlign	Выравнивание заголовка диаграммы

## 5.5 - ИЗОБРАЖЕНИЕ

Элемент для вывода изображений

autoSize	Подгонять размер под размер изображения
borderColor	Цвет границ
borderLineSize	Размер границ
borders	Границы
center	
datasource	Источник данных
field	Поле данных
format	Формат изображения
geometry	Размер и расположение объекта
geometryLocked	Блокирование размера и расположения объекта
image	Изображение
itemAlign	Выравнивание объекта
itemLocation	Расположение объекта (Страница или Бэнд)
keepAspectRatio	
objectName	Имя объекта

opacity	Прозрачность объекта
resourcePath	Путь к изображению
scale	
stretchToMaxHeight	Устанавливать значение высоты самого высокого объекта на бэнде
useExternalPainter	
variable	
watermark	Использование водяного знака

## 5.6 - ФИГУРА

Элемент для вывода фигур

borderColor	Цвет границ
borderLineSize	Размер границ
borders	Границы
cornerRadiuse	Задаёт радиус
geometry	Размер и расположение объекта
geometryLocked	Блокирование размера и расположения объекта
itemAlign	Выравнивание элемента
itemLocation	Расположение объекта (Страница или Бэнд)
lineWidth	Толщина линии
objectName	Имя объекта
opacity	Прозрачность
penStyle	Стиль линии
shape	Фигура
shapeBrush	Стиль заливки
shapeBrushColor	Цвет заливки
shapeColor	Цвет фигуры
stretchToMaxHeight	Устанавливать значение высоты самого высокого объекта на бэнде



## 5.7 - ГОРИЗОНТАЛЬНЫЙ ЛАЙОУТ

Элемент позволяющий объединять несколько элементов в группу

borderColor	Цвет границ
borderLineSize	Размер границ
borders	Границы
geometry	Размер и расположение объекта
geometryLocked	Блокирование размера и расположения объекта
hideEmptyItems	Скрывает пустые объекты
itemAlign	Выравнивание элемента
itemLocation	Расположение объекта (Страница или Бэнд)
layoutSpacing	Интервал лайоута
layoutType	Тип лайоута
objectName	Имя объекта
stretchToMaxHeight	Устанавливать значение высоты самого высокого объекта на бэнде

## 5.8 - ВЕРТИКАЛЬНЫЙ ЛАЙОУТ

Элемент позволяющий объединять несколько элементов в группу

borderColor	Цвет границ
borderLineSize	Размер границ
borders	Границы
geometry	Размер и расположение объекта
geometryLocked	Блокирование размера и расположения объекта
hideEmptyItems	Скрывает пустые объекты
itemAlign	Выравнивание элемента
itemLocation	Расположение объекта (Страница или Бэнд)
layoutSpacing	Интервал лайоута
objectName	Имя объекта
stretchToMaxHeight	Устанавливать значение высоты самого высокого объекта на бэнде

## 6 Описание функций API

**6.1 Класс ReportEngine** позволяет настраивать и использовать функции главного окна и окна предпросмотра отчета.

### Описание функций класса ReportEngine

Прототип функции	Описание
bool ReportEngine::printReport(QPrinter *printer)	Позволяет начать печать на принтере printer
bool ReportEngine::printPages(ReportPages pages, QPrinter *printer)	Печать страниц pages, на принтере printer
void ReportEngine::printToFile(const QString &fileName)	Запись в файл с именем fileName
bool ReportEngine::printToPDF(const QString &fileName)	Запись в PDF-файл с именем fileName
void ReportEngine::previewReport(PreviewHints hints)	Вывод окна для печати отчета с параметром hints. PreviewHints (файл .h относительный путь)
void ReportEngine::previewReport(QPrinter *printer, PreviewHints hints)	Вывод окна для печати отчета для принтера printer, с параметром hints
void ReportEngine::designReport()	Запуск дизайнера
PreviewReportWidget* ReportEngine::createPreviewWidget(QWidget *parent)	Создание виджета предварительного просмотра на виджете parent
void ReportEngine::setPreviewWindowTitle(const QString &title)	Устанавливает заголовок окна предварительного просмотра с названием title
void ReportEngine::setPreviewWindowIcon(const QIcon &icon)	Устанавливает иконку окна предварительного просмотра
void ReportEngine::setPreviewPageBackgroundColor(QColor color)	Устанавливает цвет заднего фона страницы color в окне предварительного просмотра
void ReportEngine::setResultEditable(bool value)	Если value — true, то включает, иначе отключает кнопку «Режим редактирования» в окне предварительного просмотра
bool ReportEngine::resultIsEditable()	Проверяет, включена ли кнопка «Режим редактирования» в окне предварительного просмотра
void ReportEngine::setSaveToFileVisible(bool value)	Если value — true, то включает, иначе отключает кнопку «Сохранить в файл» в окне предварительного просмотра
bool ReportEngine::saveToFileIsVisible()	Проверяет, включена ли кнопка «Сохранить в файл» в окне предварительного просмотра
void ReportEngine::setPrintToPdfVisible(bool value)	Если value — true, то включает, иначе отключает кнопку «Экспорт в PDF» в окне предварительного просмотра
bool ReportEngine::printToPdfIsVisible()	Проверяет, включена ли кнопка «Экспорт в PDF» в окне предварительного просмотра
void ReportEngine::setPrintVisible(bool value)	Если value — true, то включает, иначе отключает кнопку «Печати» в окне предварительного просмотра
bool ReportEngine::printIsVisible()	Проверяет, включена ли кнопка «Печати» в окне предварительного просмотра
void ReportEngine::setPreviewWindowIcon(const QIcon &icon)	Устанавливает иконку окна предварительного просмотра
bool ReportEngine::setReportLanguage(QLocale::Language language)	Присваивает отчету язык language. Возвращает true, если установленный язык language

void ReportEngine:: <b>setPreviewWindowIcon</b> (const QIcon &icon)	Устанавливает иконку окна icon предварительного просмотра
language)	поддерживается LimeReport.
QList<QLocale::Language> ReportEngine::designerLanguages()	Возвращает список доступных языков для отчета
QLocale::Language ReportEngine::currentDesignerLanguage()	Возвращает текущий язык отчета
void ReportEngine::addWatermark(const WatermarkSetting &watermarkSetting)	Добавляет водяной знак с настройками watermarkSetting
void ReportEngine::clearWatermarks()	Удаляет водяные знаки
IPreparedPages *ReportEngine::preparedPages()	Возвращает готовые страницы отчета
QString ReportEngine::reportFileName()	Возвращает имя отчета
void ReportEngine::setReportFileName(const QString &fileName)	Устанавливает имя файла fileName для отчета
String ReportEngine::lastError()	Возвращает последнюю ошибку
void ReportEngine::setCurrentReportsDir(const QString &dirName)	Устанавливает текущую директорию dirName для отчета
void ReportEngine::setReportName(const QString &name)	Устанавливает имя name отчету
QString ReportEngine::reportName()	Возвращает название отчета
void ReportEngine::cancelRender()	Отменяет отрисовку отчета

**6.2 Интерфейс IDataSourceManager** позволяет использовать источники данных и обрабатывать переменные.

Описание функций интерфейс IDataSourceManager

Прототип функции	Описание
void <b>setReportVariable</b> (const QString& name, const QVariant& value);	Устанавливает переменной отчета name, значение value. Если переменная name существует, её значение value перезаписывается
void <b>setDefaultDatabasePath</b> (const QString &defaultDatabasePath);	Устанавливает путь defaultDatabasePath к базе данных
bool <b>containsVariable</b> (const QString& variableName);	Возвращает true, если переменная с именем variableName существует
QVariant <b>variable</b> (const QString& variableName);	Возвращает переменную с именем variableName
bool <b>addModel</b> (const QString& name, QAbstractItemModel *model, bool owned);	Подключает внешний источник данных, реализующий QAbstractItemModel. При параметре owned = true, LimeReport удаляет модель
void <b>removeModel</b> (const QString& name);	Удаление модели с именем name
bool <b>containsDatasource</b> (const QString& dataSourceName);	Возвращает true, если источник данных с именем dataSourceName существует
void <b>clearUserVariables</b> ();	Удаляет учетные данные базы данных
ICallbackDatasource* <b>createCallbackDatasource</b> (const QString& name);	Создает новый источник обратного вызова данных с именем name
QStringList <b>variableNames</b> ();	Возвращает список имен переменных

Прототип функции	Описание
bool <b>variableIsMandatory</b> (const QString& name);	Возвращает true, если для переменной name установлен параметр Mandatory
VariableDataType <b>variableDataType</b> (const QString& name);	Возвращает тип переменной name
bool <b>variableIsSystem</b> (const QString& name);	Возвращает true, если тип переменной name – VarDesc::System
IDataSource* <b>dataSource</b> (const QString& name);	Возвращает источник данных с именем name.

### 6.3 Интерфейса IDbCredentialsProvider возвращает учетные данные базы данных.

#### Описание функций интерфейса IDbCredentialsProvider

Прототип функции	Описание
QString <b>getUserName</b> (const QString& connectionName);	Возвращает имя пользователя из источника данных connectionName
QString <b>getPassword</b> (const QString& connectionName);	Возвращает пароль пользователя из источника данных connectionName

### 6.4 Интерфейс IDataSource работает с моделью данных.

#### Описание функций интерфейса IDataSource

Прототип методов	Описание
bool <b>next</b> ();	Возвращает true, если указатель текущей записи имеет ссылку на следующую запись и указатель переходит к ней
bool <b>hasNext</b> ();	Возвращает true, если указатель текущей записи имеет ссылку на следующую запись
bool <b>prior</b> ();	Возвращает true, если указатель текущей записи имеет ссылку на предыдущую запись и указатель переходит на предыдущую запись
void <b>first</b> ();	Возвращает true, если указатель текущей записи расположен на первой записи набора данных
void <b>last</b> ();	Возвращает true, если указатель текущей записи расположен на последней записи набора данных
bool <b>bof</b> ();	Возвращает true, если указатель текущей записи расположен перед первой записью набора данных
bool <b>eof</b> ();	Возвращает true, если указатель текущей записи расположен после последнего набора данных
QVariant <b>data</b> (const QString& columnName);	Возвращает данные столбца columnName
QVariant <b>dataByRowIndex</b> (const QString& columnName, int rowIndex);	Возвращает данные находящиеся в столбце columnName и строке rowIndex
QVariant <b>dataByKeyField</b> (const QString& columnName, const QString& keyColumnName, QVariant keyData);	Возвращает ключ записи находящейся в столбце columnName, с ключем столбца keyColumnName и ключевыми данными keyData
int <b>columnCount</b> ();	Возвращает количество столбцов
QString <b>columnNameByIndex</b> (int columnIndex);	Возвращает название столбца по его индексу columnIndex

Прототип методов	Описание
int <b>columnIndexByName</b> (QString name);	Возвращает индекс столбца по его названию name
bool <b>isInvalid</b> ();	Возвращает true, если модель не существует
QString <b>lastError</b> ();	Возвращает последнюю ошибку
QAbstractItemModel* <b>model</b> ();	Возвращает модель

## 6.5 Интерфейс **IDataSourceHolder** работает с моделью данных.

### Описание функций интерфейса **IDataSourceHolder**

Прототип функции	Описание
QString <b>lastError</b> ();	Возвращает последнюю ошибку
bool <b>isInvalid</b> ();	Возвращает true, если источник данных недействителен
bool <b>isOwned</b> ();	Возвращает true, если источник данных принадлежит
bool <b>isEditable</b> ();	Возвращает true, если источник данных можно редактировать
bool <b>isRemovable</b> ();	Возвращает true, если источник данных можно удалить
void <b>invalidate</b> (IDataSource::DataSourceMode mode, bool dbWillBeClosed = false);	Аннулирует источник данных с модом mode
void <b>update</b> ();	Обновляет источник данных
void <b>clearErrors</b> ();	Очищает хранящиеся ошибки

## 6.6 Интерфейс **IPreparedPages** подготавливает страницы отчета.

### Описание функций интерфейса **IPreparedPages**

Прототип функции	Описание
bool <b>loadFromFile</b> (const QString& fileName);	Возвращает true, если данные были загружены из файла с именем fileName
bool <b>loadFromString</b> (const QString data);	Возвращает true, если данные были загружены из строки data
bool <b>loadFromByteArray</b> (QByteArray* data);	Возвращает true, если данные были загружены из QByteArray с именем data
bool <b>saveToFile</b> (const QString& fileName);	Возвращает true, если данные были сохранены в файле с именем fileName
QString <b>saveToString</b> ();	Сохраняет строку
QByteArray <b>saveToByteArray</b> ();	Сохраняет QByteArray
void <b>clear</b> ();	Очищает список подготовленных страниц

## 6.7 Класс **PreviewReportWidget** работает с виджетом предпросмотра отчета.

### Описание функций класса **PreviewReportWidget**

Прототип функции	Описание
explicit <b>PreviewReportWidget</b> (ReportEngine *report, QWidget *parent = 0);	Создает объект класса PreviewReportWidget от объекта report класса ReportEngine, на виджете parent
bool <b>exportReport</b> (QString exporterName, const QMap<QString, QVariant>& params = QMap<QString, QVariant>());	Экспортирует отчет с именем exporterName и параметрами params
ScaleType <b>scaleType</b> () const;	Возвращает тип шкалы
int <b>scalePercent</b> () const;	Возвращает процент шкалы

Прототип функции	Описание
void <b>setScaleType</b> (const ScaleType &scaleType, int percent = 0);	Устанавливает тип шкалы scaleType
void <b>setPreviewPageBackgroundColor</b> (QColor color);	Устанавливает цвет color для фона страницы предпросмотра
QColor <b>previewPageBackgroundColor</b> ();	Возвращает цвет фона страницы предпросмотра
QPrinter * <b>defaultPrinter</b> () const;	Возвращает принтер, установленный по умолчанию
void <b>setDefaultPrinter</b> (QPrinter *defaultPrinter);	Устанавливает принтер defaultPrinter по умолчанию
void <b>startInsertTextItem</b> ();	Начинает вставлять текст в объект
void <b>activateItemSelectionMode</b> ();	Активирует режим выбора объекта
void <b>deleteSelectedItems</b> ();	Удаляет выбранные объекты

## 6.8 Класс PrintRange реализует диапазон печати.

### Описание функций класса PrintRange

Прототип функции	Описание
int <b>fromPage</b> () const { return m_fromPage;}	Печать со страницы m_fromPage
int <b>toPage</b> () const { return m_toPage;}	Печать до страницы m_toPage
QPrintDialog::PrintRange <b>rangeType</b> () const { return m_rangeType;}	Возвращает тип диапазона
void <b>setRangeType</b> (QAbstractPrintDialog::PrintRange rangeType){ m_rangeType=rangeType;}	Устанавливает тип диапазона rangeType
void <b>setFromPage</b> (int fromPage){ m_fromPage = fromPage;}	Устанавливает с какой страницы fromPage необходимо начать печать
ReportSettings * <b>reportSettings</b> ();	Возвращает настройки отчета
void <b>setReportSettings</b> (ReportSettings *reportSettings);	Устанавливает настройки отчета reportSettings
void <b>setToPage</b> (int toPage){ m_toPage = toPage;}	Устанавливает до какой страницы toPage необходимо печатать

## 6.9 Интерфейс ReportDesignWindowInterface обрабатывает разные настройки дизайнера LimeReport.

### Описание функций интерфейса ReportDesignWindowInterface

Прототип функции	Описание
bool <b>checkNeedToSave</b> ();	Проверяет сохранен ли отчет перед выходом из дизайнера. Если отчет не сохранен, то появится предупреждение с предложением сохранить отчет
void <b>setSettings</b> (QSettings* value);	Устанавливает настройки value для дизайнера
QSettings* <b>settings</b> ();	Возвращает настройки дизайнера
void <b>restoreSetting</b> ();	Восстанавливает настройки

**6.10 Класс ItemGeometry** устанавливает геометрию для объекта, в частности: положение, размер, тип, выравнивание.

Описание функций класса ItemGeometry

Прототип функции	Описание
qreal <b>x</b> () const;	Возвращает положение объекта на оси x
void <b>setX</b> (const qreal &x);	Устанавливает положение объекта на оси x
qreal <b>y</b> () const;	Возвращает положение объекта на оси y
void <b>setY</b> (const qreal &y);	Устанавливает положение объекта на оси y
qreal <b>width</b> () const;	Возвращает ширину объекта
void <b>setWidth</b> (const qreal &width);	Устанавливает ширину объекта width
qreal <b>height</b> () const;	Возвращает высоту объекта
void <b>setHeight</b> (const qreal &height);	Устанавливает высоту объекта
Type <b>type</b> () const;	Возвращает тип объекта
void <b>setType</b> (const Type &type);	Устанавливает тип объекта type
Qt::Alignment <b>anchor</b> () const;	Возвращает выравнивание объекта
void <b>setAnchor</b> (const Qt::Alignment &anchor);	Устанавливает выравнивание объекта anchor

**6.11 Класс WatermarkSetting** отвечает за настройки водяного знака.

Описание функций класса WatermarkSetting

Прототип функции	Описание
<b>WatermarkSetting</b> (const QString& text, const ItemGeometry& geometry, const QFont& font) : m_text(text), m_font(font), m_opacity(50), m_geometry(geometry), m_color(QColor(Qt::black)){}	Создает объект класса WatermarkSetting с расположением и размером geometry и шрифтом font. Также настройкам водяного знака присваивается текст text, шрифт font, прозрачность 50%, расположение и размер geometry, цвет черный.
<b>WatermarkSetting</b> () : m_font(QFont()), m_opacity(50), m_geometry(ItemGeometry()) {}	Создает объект класса WatermarkSetting. Также настройкам водяного знака присваивает шрифт font, прозрачность 50%, расположение и размер geometry.
QString <b>text</b> () const;	Возвращает текст водяного знака
void <b>setText</b> (const QString &text);	Устанавливает текст text для водяного знака
QFont <b>font</b> () const;	Возвращает шрифт водяного знака
void <b>setFont</b> (const QFont &font);	Устанавливает шрифт font для водяного знака
int <b>opacity</b> () const;	Возвращает прозрачность водяного знака
void <b>setOpacity</b> (const int &opacity);	Устанавливает прозрачность для водяного знака
ItemGeometry <b>geometry</b> () const;	Возвращает расположение и размер водяного знака
void <b>setGeometry</b> (const ItemGeometry &geometry);	Устанавливает расположение и размер geometry водяного знака
QColor <b>color</b> () const;	Возвращает цвет водяного знака

<code>void setColor(const QColor &amp;color);</code>	Устанавливает цвет color для водяного знака
--	---

**6.12 Класс ReportError** наследован от `std::runtime_error` и может возвращать ошибки вызванными исключениями.

Описание функций класса ReportError

Прототип функции	Описание
<code>ReportError(const QString&amp; message);</code>	Возвращает ошибку message.

**6.13 Класс ReportSettings** содержит одну настройку для ПС LimeReport, которая позволяет скрывать предупреждения об отсутствии полей и переменных.

Описание функций класса ReportSettings



Прототип функции	Описание
<code>void setDefaultValues()</code>	Устанавливает значение настроек по умолчанию. По умолчанию LimeReport показывает все предупреждения
<code>bool suppressAbsentFieldsAndVarsWarnings(const);</code>	Возвращает true, если LimeReport не показывает предупреждения об отсутствии полей и переменных
<code>void setSuppressAbsentFieldsAndVarsWarnings(bool suppressAbsentFieldsAndVarsWarnings);</code>	Устанавливает настройки показа предупреждений об отсутствии полей и переменных. Если <code>suppressAbsentFieldsAndVarsWarnings</code> имеет значение true, то предупреждения об отсутствии полей и переменных показываться не будут, если false, то предупреждения об отсутствии полей и переменных будут показываться



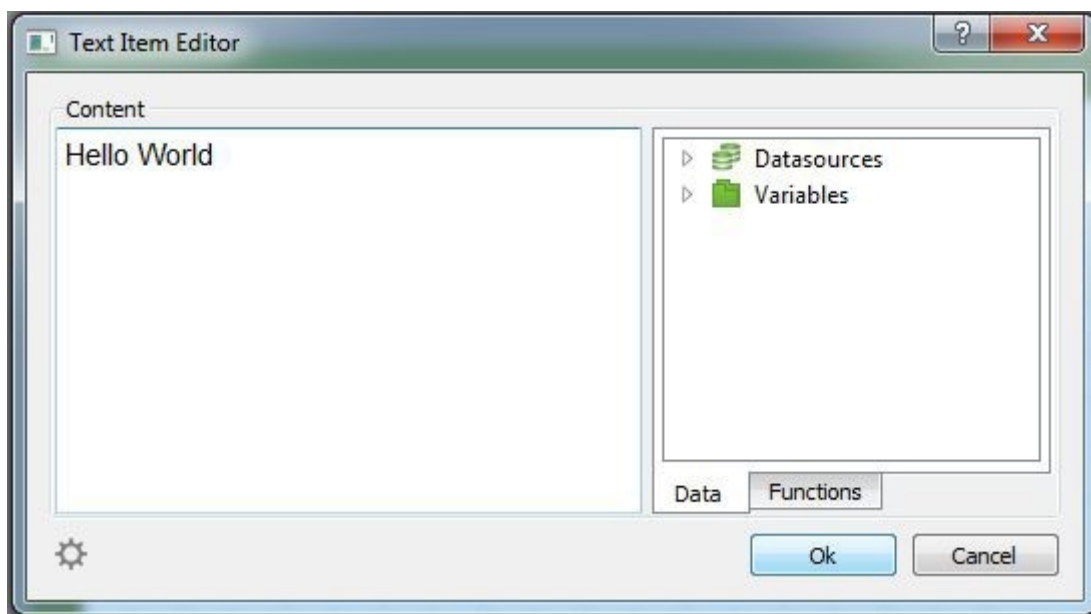
## 7 ПОСТРОЕНИЕ ОТЧЕТОВ

### 7.1 Hello World

По традиции самый первый и простой отчет “Hello World”


Создаем новый отчет . Кликаем на объект “Текст”  на панели объектов.

Перемещаем указатель мыши на нужное место на листе, и опять нажимаем клавишу мыши. Дважды кликаем на только что созданном объекте, тем самым активировав окно редактора.



Набираем текст “Hello World” и нажимаем кнопку ОК



Отчет готов. Для его просмотра можно воспользоваться кнопкой предварительного  просмотра на панели инструментов или пунктом меню File | Render Report, также можно воспользоваться сочетанием клавиш Ctrl+P. Вы увидите окно предварительного просмотра с единственной страницей отчета, которая содержит надпись "Hello World!".

## 7.2 Объект текст

Объект "Текст" обладает очень широкими возможностями. Он умеет отображать текст, рамку, заливку. Текст может быть отображен любым шрифтом, любого размера, цвета и стиля. Большинство настроек делаются визуально с помощью панелей инструментов.

### Приметы оформления текста


Hello World	Hello World	Hello World
Hello World	<b>Hello World</b>	<i>Hello World</i>
Hello World	<u>Hello World</u>	rlroM ollɹɹ

Для знакомства с объектом "Текст" создадим его и разместим в нем некоторый текст:

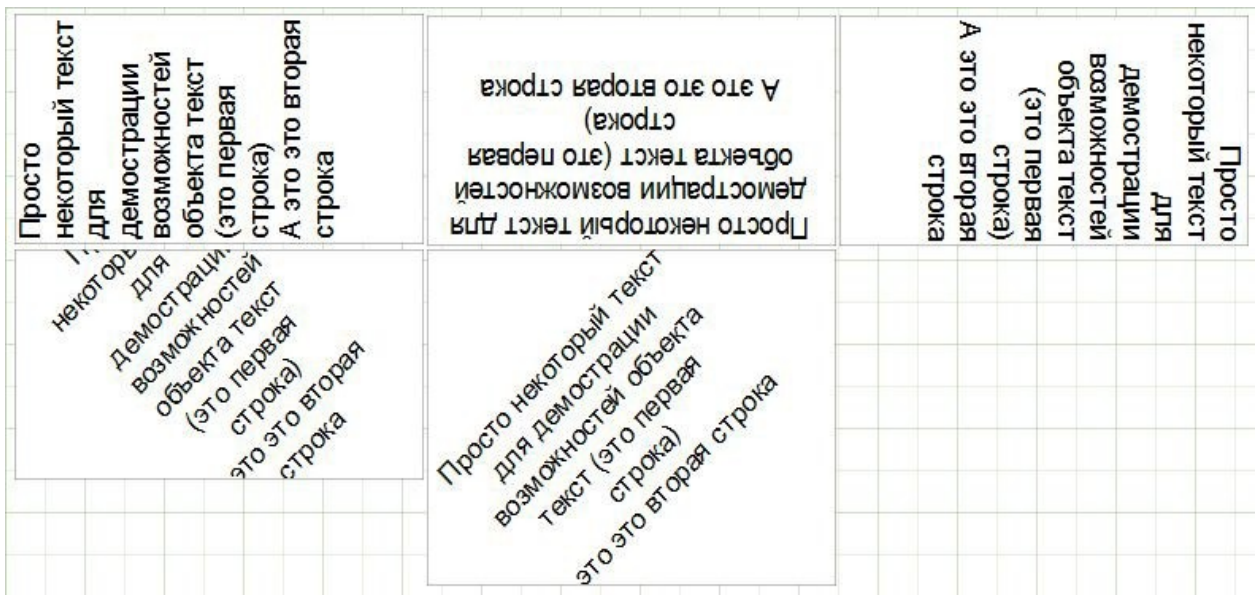
*Это первая строка  
демонстрационного текста  
А это вторая*

Теперь посмотрим, как работает выравнивание текста внутри объекта.



Кнопки выравнивания расположены на панели инструментов и позволяют независимо задать выравнивание текста по горизонтали и по вертикали. Обратите внимание на кнопку "Выравнивание по ширине"  - она позволяет выровнять параграф по обоим краям объекта.

Просто некоторый текст для демонстрации возможностей объекта текст (это первая строка) А это это вторая строка	Просто некоторый текст для демонстрации возможностей объекта текст (это первая строка) А это это вторая строка	Просто некоторый текст для демонстрации возможностей объекта текст (это первая строка) А это это вторая строка
Просто некоторый текст для демонстрации возможностей объекта текст (это первая строка) А это это вторая строка	Просто некоторый текст для демонстрации возможностей объекта текст (это первая строка) А это это вторая строка	Просто некоторый текст для демонстрации возможностей объекта текст (это первая строка) А это это вторая строка



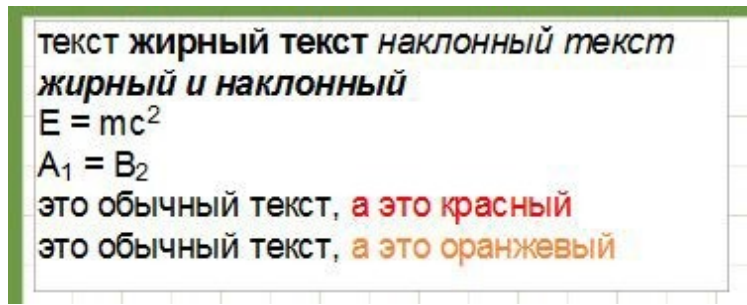
Весь текст может быть повернут на 45, 90, 180, 270 и 315 градусов. При повороте на значения, отличные от 90, 180, 270, текст может вылезти за пределы объекта, как в нашем случае (см. рис.). Чтобы текст полностью уместился, немного увеличим высоту объекта.

Объект "Текст" понимает HTML теги Например:

```

текст <b>жирный текст</b> <i>наклонный текст <b>жирный и
наклонный</i></b> E = mc<sup>2</sup>
A<sub>1</sub> = B<sub>2</sub>
это обычный текст,<font color=red> а это красный </font>
это обычный текст,<font color = #ff8030> а это оранжевый</font>

```



Отображение выражений с помощью объекта "Текст"

Одна из самых главных особенностей этого универсального объекта - это возможность отображения не только статического текста, но и выражений. Причем, выражения могут располагаться в объекте вперемешку с текстом. Рассмотрим простой пример - поместим в объект "Текст" следующую строку:

```
Привет! Сегодня $$now()
```


Если запустить отчет на построение, мы увидим приблизительно следующее:

*Привет! Сегодня 2015-08-03*

Что произошло? В процессе построения отчета LimeReport встретил в тексте выражение, заключенное  $\$\${\}$ , вычислил его и вставил полученное значение обратно в текст, убрав, разумеется, скобки. Объект "Текст" может содержать любое количество выражений, смешанных с обычным текстом. В скобки можно заключать и одиночные переменные, и выражения, например,  $\$\${1+2*(3+4)}$ . В выражениях можно использовать константы, переменные ( $\$V{\}$ ), функции, поля БД ( $\$D{\}$ ).

### 7.3 Использование бэндов

Бэнды применяются для логической группировки объектов. Так, разместив объект на бэнде типа "Page Header", мы тем самым говорим LimeReport, что данный объект надо вывести на каждой странице готового отчета вверху. Аналогичным образом бэнд "Page Footer" выводится внизу каждой страницы, со всеми лежащими на нем объектами. Продемонстрируем это небольшим примером. Сделаем отчет, который содержит надпись "Hello!" вверху страницы, текущую дату вверху справа и номер страницы внизу справа.

Для этого на панели объектов щелкните кнопку "Вставить бэнд"  и из открывшегося списка выберите "Page Header". Мы видим, что на страницу добавился новый бэнд. Дизайнер LimeReport автоматически размещает бэнды на странице таким образом, чтобы вверху находились бэнды-заголовки, после них - бэнды-данные, и ниже всех - бэнды-завершения.

Далее мы вставляем бэнды "Report Header" и "Page Footer"

Теперь размещаем объекты. На бэнд "Page Header" помещаем объект "Текст" и в его редакторе набираем  $\$\${now()}$ . На бэнд "Report Header" помещаем объект "Текст", который будет содержать текст "Hello!". А на бэнде "Page Footer", мы разместим объект текст в котом будет использована системная переменная содержащая номер страницы  $\$V{\#PAGE}$



Запустим отчет на выполнение и увидим, что объекты в готовом отчете разместились именно так, как нам нужно.

2015-08-04



Hello World

Итак, за размещение объектов в нужном месте отчета отвечают бэнды. В зависимости от типа бэнда мы можем расположить объект вверху или внизу страницы, на первой странице, на последней странице. Основные бэнды, которые могут нам понадобиться в большинстве отчетов, работают следующим образом: - бэнд "Page Header" выводится в самом верху на каждой странице; - бэнд "Page Footer" выводится в самом низу на каждой странице; - бэнд "Report Header" выводится на первой странице отчета вверху, но после бэнда "Page Header" "Report Footer" выводится в самом конце отчета, на свободном месте

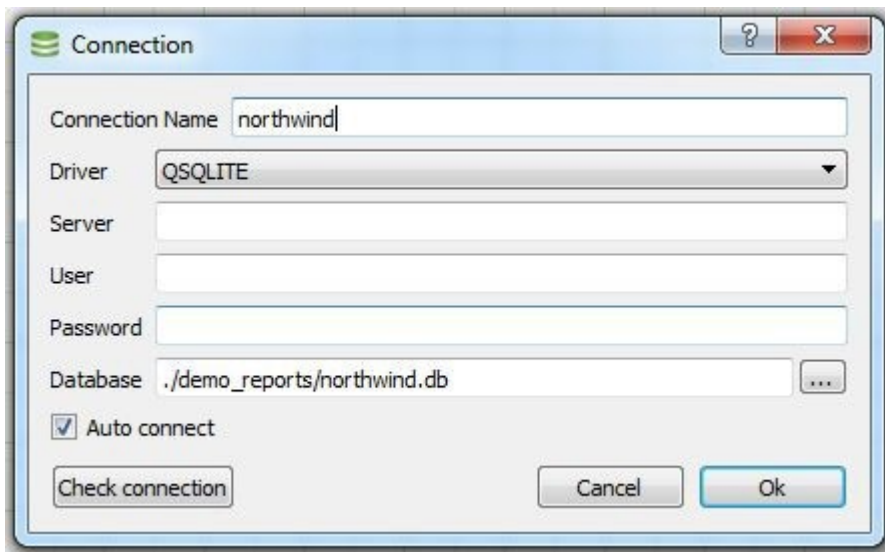
## 7.4 Бэнды-данные


Далее мы рассмотрим возможности выводить на печать данные из наборов данных. Что такое набор данных в данном случае? Это заранее неизвестное количество строк (записей), каждая из которых содержит определенное количество колонок (полей). Для печати такого рода информации LimeReport использует особый тип бэндов - бэнды-данные, или дата-бэнды. Это бэнды с названиями "Data" и "SubDetail". Чтобы напечатать всю таблицу или некоторые ее поля, необходимо: - добавить дата-бэнд в отчет; - подключить его к таблице; - разместить на нем объекты "Текст" с полями, которые мы хотим распечатать. При построении отчета LimeReport повторит печать бэнда столько раз, сколько записей в нашей таблице. При этом, если закончилось свободное место на странице, будут сформированы новые страницы отчета

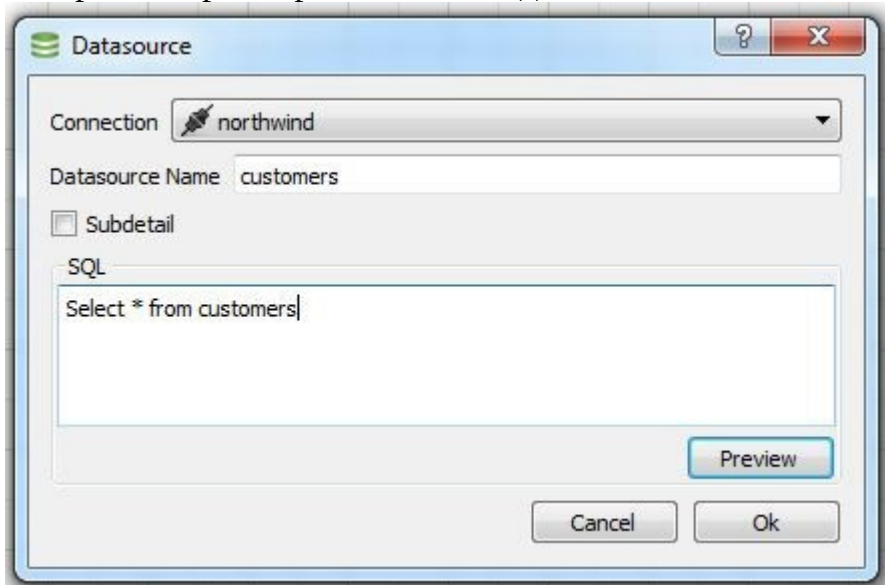
## 7.5 Отчет "Список клиентов"

Данный отчет будет содержать данные из таблицы клиентов из демонстрационной БД "northwind". Для начала нажмем кнопку новый отчет . Далее мы создадим подключение к демонстрационной базе данных. Для этого мы воспользуемся кнопкой  (Add database connection), расположенной на панели инструментального окна "Data Browser". В открывшемся диалоге "Connection" настроим параметры подключения.







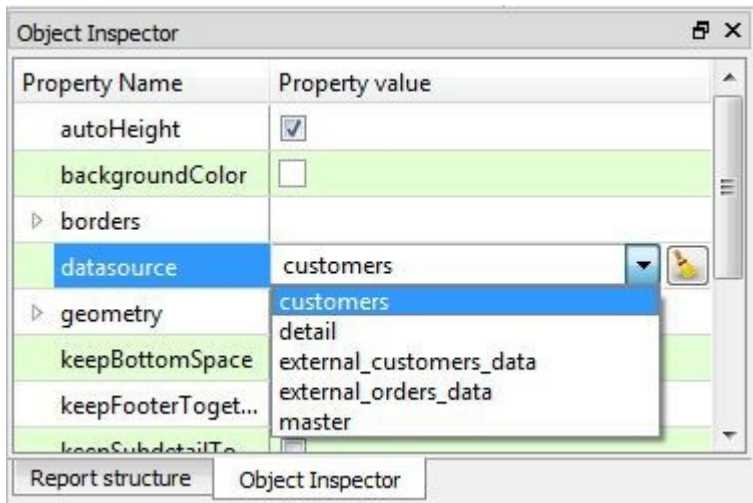
Далее создадим набор данных “Customers” основанный на SQL запросе. Для этого воспользуемся кнопкой  расположенной на панели инструментального окна “Data Browser”. Используя диалог “Datasource” настроим параметры источника данных



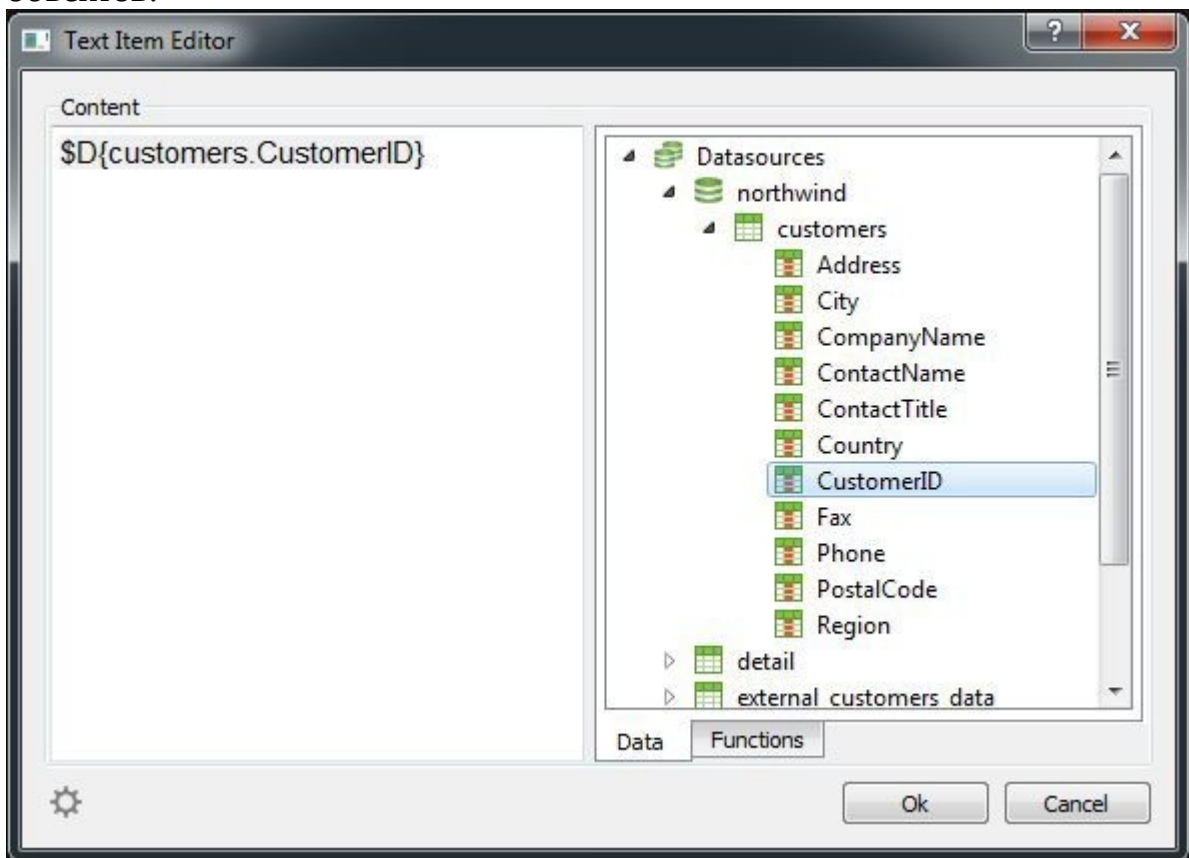
Connection - наименование подключения (Выбираем ранее созданное подключение “northwind”)

Datasource Name - наименование набора данных (Устанавливаем в “customers”) SQL - текст SQL запроса (Набираем Select \* from customers)

Теперь можно приступить к созданию формы отчета. Для этого мы с помощью кнопки  (Вставить бэнд) добавляем к отчету бэнд “Report Header” и размещаем на нем объект “Текст”  содержащий текст “Список клиентов”. Далее добавляем “Data” бэнд. Устанавливаем с помощью “Обозревателя объектов” в свойство “datasource” значение “customers”



Теперь разместим на бэнде четыре объекта, которые будут отображать номер клиента, его наименование, телефон и факс. Сделаем это разными способами, чтобы продемонстрировать широкие возможности дизайнера LimeReport. Первый объект "Текст" положим на бэнд и наберем в нем текст "\$D{customers.CustomerID}". Это самый неудобный способ, т.к. приходится ссылку на поле писать вручную, и мы можем легко ошибиться. Чтобы облегчить вставку таких ссылок в текст, можно воспользоваться деревом источников данных расположенным в правой части редактора текстовых объектов.





Раскрыв дерево и найдя нужное нам поле осуществим двойной клик на нем и в редактор слева в текущую позицию курсора вставится нужное нам выражение `$D{customers.CustomerID}`

Второй способ - drag&drop нужного поля из служебного окна "Data Browser" в отчет. Это самый простой и наглядный способ. Схватите мышкой поле "Phone" и перетащите его на бэнд.

The screenshot shows the 'Data Browser' window with a tree view of fields: CustomerID, Fax, Phone, PostalCode, and Region. The 'Phone' field is selected. Below it, a report design grid is shown with a table structure. The table has columns for CustomerID, CompanyName, and Phone. The grid also includes ReportHeader1, PageFooter1, and a page number field \$V{#PAGE}.

Итак наш отчет готов

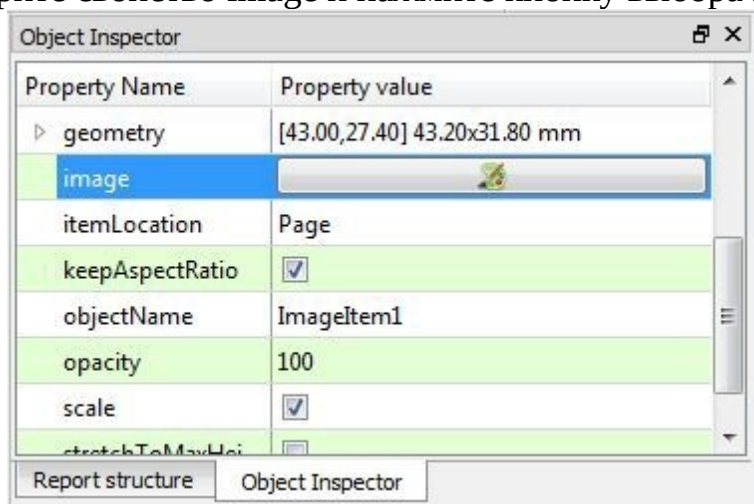
Нажимаем на предварительный просмотр и видим что получилось.

Список клиентов			
ALFKI	Alfreds Futterkiste	030-0074321	030-0076545
ANATR	Ana Trujillo Emparedados y helados	(5) 555-4729	(5) 555-3745
ANTON	Antonio Moreno Taqueraa	(5) 555-3932	
AROUT	Around the Horn	(71) 555-7788	(71) 555-6750
BERGS	Berglunds snabbkap	0921-12 34 65	0921-12 34 67
BLAUS	Blauer See Delikatessen	0621-08460	0621-08924
BLONP	Blondel pare et fils	88.60.15.31	88.60.15.32
BOLID	Balido Comidas preparadas	(91) 555 22 82	(91) 555 91 99

## 7.6 Объект "Рисунок"

Следующий объект, который мы рассмотрим - это объект "Рисунок". Он также довольно часто используется в отчетах. С помощью объекта вы можете вставить в отчет логотип вашей фирмы, фотографию сотрудника или любую другую графическую информацию.

Давайте рассмотрим возможности объекта. Создайте пустой отчет и поместите на лист отчета объект "Рисунок". В обозревателе объектов выберите свойство `image` и нажмите кнопку выбора и загрузки изображения.




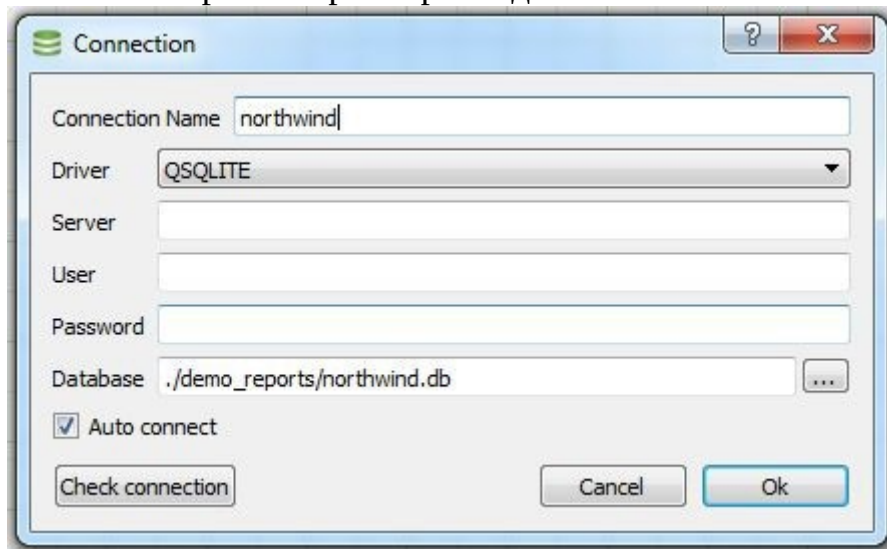
В обозревателе объекта мы также можем наблюдать следующие опции : `autoSize`, `scale` (включено по умолчанию), `center`(включено по умолчанию), `keepAspectRatio`,(включено по умолчанию). Включив опцию "Автора размер" мы увидим, что объект принял размеры, соответствующие находящемуся в нем рисунку. Иногда такая возможность бывает полезна, если надо отображать рисунки разных размеров. По умолчанию эта опция выключена, что подходит для большинства случаев. Опция "`scale`" включена по умолчанию, что заставляет рисунок растягиваться внутри объекта. Изменяйте размеры объекта мышкой, и вы увидите, что размер картинки все время соответствует размеру объекта. Если опцию отключить, то рисунок будет отображаться в исходных размерах. Это поведение отличается от опции "`autoSize`" тем, что размеры объекта не подгоняются под размер рисунка, т.е. объект можно сделать больше рисунка или меньше. Опция "`center`" позволяет отцентрировать рисунок внутри объекта. Опция "`keepAspectRatio`" включена по умолчанию и выполняет очень полезную задачу: не позволяет пропорциям рисунка искажаться при изменении размеров объекта. Эта опция работает только в паре с опцией "Растягивание". При любом изменении размеров объекта нарисованный круг останется кругом, а не превратится в овал. При этом растянутый рисунок занимает не весь внутренний объем объекта, а только часть, необходимую для отображения картинки в правильных пропорциях. Если опцию отключить, то картинка растянется на весь объем объекта, и, если размеры объекта не соответствуют исходным пропорциям картинки, картинка исказится

## 7.7 Отчет с картинками

Объект "Рисунок", как и другие объекты в LimeReport, умеет отображать данные из БД. Подключение объекта к нужному полю БД осуществляется с помощью свойств `datasource`, `field` в обозревателе объектов. В отличие от объекта "Текст", это единственный способ подключить объект к данным. Продемонстрируем все вышесказанное примером отчета, который будет содержать изображения категорий товаров вместе с их названиями. Для этого нам опять потребуется демонстрационная база данных `northwind`, идущая в комплекте с LimeReport.

Зайдем в дизайнер и нажмем кнопку "Новый отчет", чтобы LimeReport автоматически создал пустой шаблон.

Подключим таблицу к отчету в окне. Для этого мы воспользуемся кнопкой  (Add database connection), расположенной на панели инструментального окна "Data Browser". В открывшемся диалоге "Connection" настроим параметры подключения.

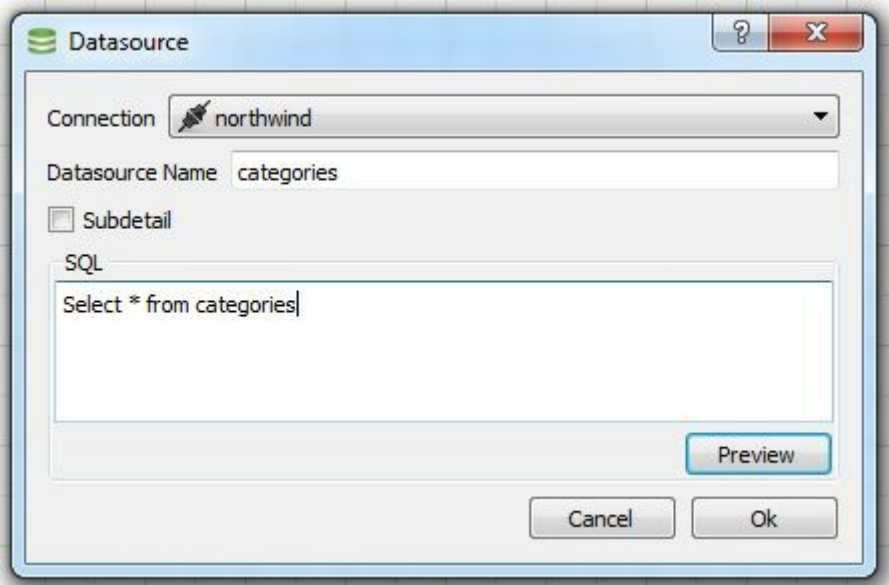


Connection Name - наименование подключения Driver - драйвер для подключения к БД (SQLite)

DataBase - путь к базе данных

AutoConnect - автоматически подключаться после создания описателя подключения и загрузки отчета.



Далее создадим набор данных "Categories" основанный на SQL запросе. Для этого воспользуемся кнопкой  расположенной на панели инструментального окна "Data Browser". Используя диалог "Datasource" настроим параметры источника данных

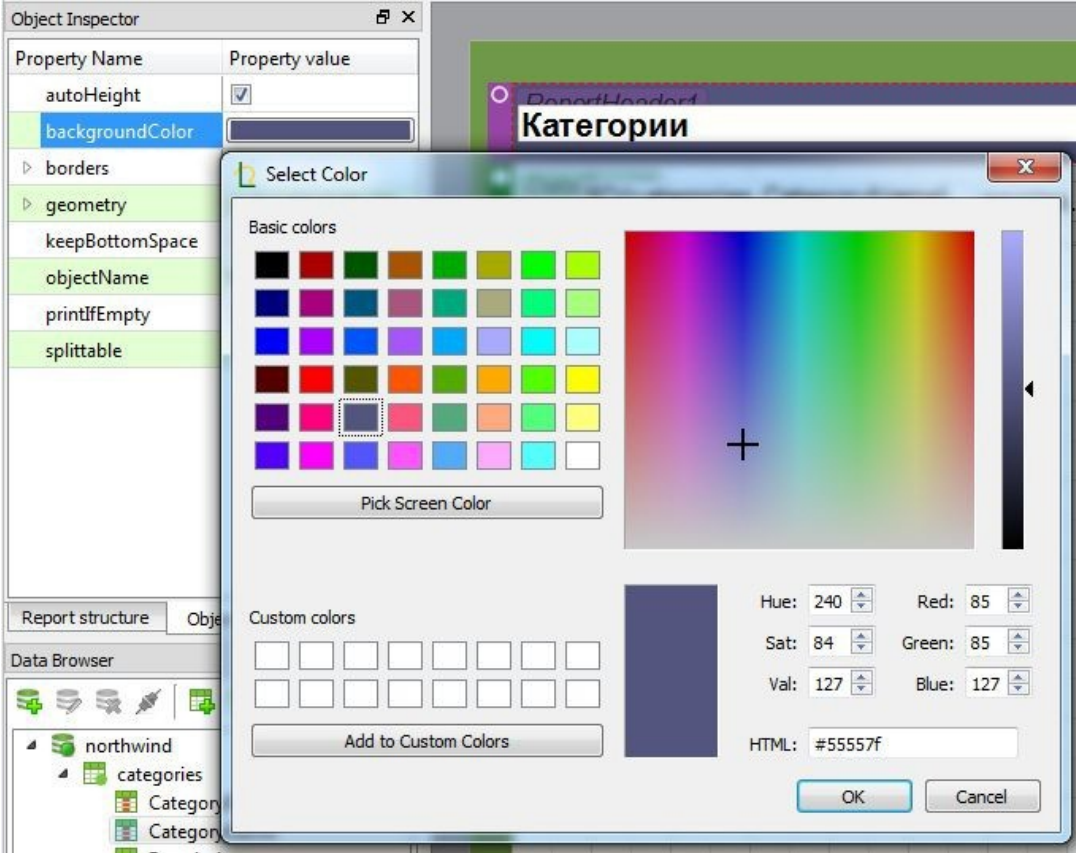


Connection - наименование подключения (Выбираем ранее созданное подключение “northwind”)

Datasource Name - наименование набора данных (Устанавливаем в “categories”)

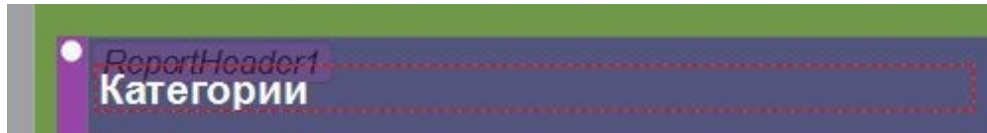
SQL - текст SQL запроса (Набираем Select \* from categories)



Теперь можно приступить с созданию формы отчета. Для этого мы с помощью кнопки  (Вставить бэнд) добавляем к отчету бэнд “Report Header” и размещаем на нем объект “Текст”  содержащий текст “Категории”. Далее выберем цвет заливки бэнда воспользовавшись свойство backgroundColor у объекта бэнд.





Как мы можем заметить объект "Текст" сильно выделяется на фоне заливки бэнда. Для того чтобы это исправить мы поменяем свойства `fontColor` и `backgroundMode` у этого объекта. Устанавливаем "backgroundMode" в "TransparentMode" а `fontColor` устанавливаем в белый.

Таким образом получаем белую надпись на прозрачном фоне.

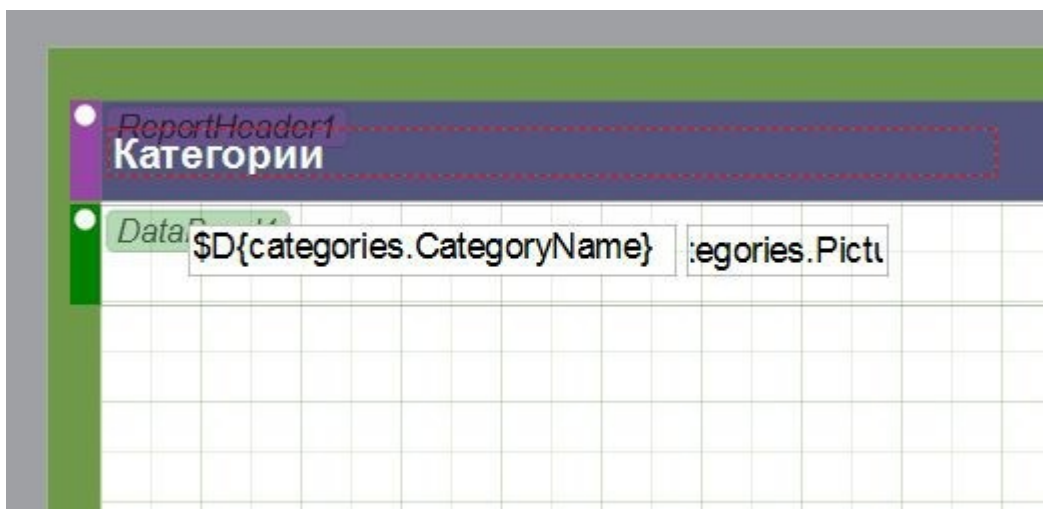


Далее добавляем  "Data" бэнд. Устанавливаем с помощью "Обозревателя объектов" в свойство "datasource" значение "categories". На бэнд положим объект "Текст"  и подключим его к полю `SD{categories.CategoryName}` одним из способов, описанных выше.

Рядом разместим объект "Рисунок"  и подключим его к полю "Picture".

datasource	categories
field	Picture 
geometry	CategoryID CategoryName Description
image	Picture
itemLocation	band
keepAspectRatio	<input checked="" type="checkbox"/>
objectName	ImageItem1

Для этого в обозревателе объектов настроим свойства: `datasource = "categories"`, `field = "Picture"` напомним, что оба этих свойства - типа "список", поэтому нужные значения можно выбрать с помощью мыши. А так же отметим галочкой параметр `autoSize`.





## Категории

Beverages



Condiments



Confections



## 7.8 Отображение многострочного текста

Вернемся к предыдущему примеру с категориями. В таблице “categories” есть поле “Description”, которое содержит описание каждой категории. Давайте модернизируем наш отчет, добавив в него это поле. На первый взгляд все просто: добавляем на бэнд с данными объект “Текст”, подключаем его к полю “Description” и устанавливаем размеры объекта.

Запускаем отчет на выполнение и видим, что получилось не совсем то, чего мы ожидали:

Категории		
Beverages	Drinks, or beverages, are liquids intended for human consumption. In addition to basic	
Condiments	A condiment is a spice, sauce or other food preparation that is added to food to impart a	
Confections	Sugar confections include sweet, sugar-based foods, which are usually eaten as snack food.	

Однако, LimeReport всего лишь сделал то, что его просили сделать. Поле “Description” содержит многострочный текст, размер которого может варьироваться. А наш объект “Текст”, отображающий информацию из этого поля, имеет фиксированный размер. Вот некоторые строки и не влезли в объект и были обрезаны. Как поступить в данной ситуации? Можно, конечно, подобрать размеры объекта с запасом или уменьшить размер шрифта. Однако, это приведет к неэкономному использованию места на листе: одни категории имеют длинное описание, другие - короткое. В LimeReport есть средства, позволяющие решить эту проблему. Речь идет о возможности бэнда подбирать свою высоту таким образом, чтобы вместить все имеющиеся в нем объекты. Для этого надо всего лишь включить свойство “autoHeight”. Однако это не все - объект с длинным текстом и сам должен уметь растягиваться. Объект “Текст” умеет это делать. Объект может автоматически подбирать свою высоту или ширину, чтобы полностью вместить имеющийся в нем текст. Для этого служат свойства “autoWidth” и “autoHeight”. Свойство “autoWidth” подбирает ширину объекта таким образом, чтобы уместились все строки, без переносов слов. Этот режим удобен, когда объект содержит единственную строку текста. Свойство “autoHeight” позволяет подобрать высоту объекта так, чтобы поместился весь текст. Ширина объекта при этом не меняется. Включите свойство “autoHeight” у объекта содержащего поле “Description”. Также включите свойство “autoHeight” у бэнда.

Запустим отчет и убедимся, что теперь все работает как надо.

## Категории

### Beverages

Drinks, or beverages, are liquids intended for human consumption. In addition to basic needs, beverages form part of the culture of human society. Although all beverages, including juice, soft drinks, and carbonated drinks, have some form of water in them, water itself is often not classified as a beverage, and the word beverage has been recurrently defined as not referring to water .



An alcoholic beverage is a drink containing ethanol, commonly known as alcohol, although in chemistry the definition of an alcohol includes many other compounds. Alcoholic beverages, such as wine, beer, and liquor, have been part of human culture and development for 8,000 years.

Non-alcoholic beverages often signify drinks that would normally contain alcohol, such as beer and wine but are made with less than .5 percent alcohol by volume. The category includes drinks that have undergone an alcohol removal process such as non-alcoholic beers and de-alcoholized wines.

### Condiments

A condiment is a spice, sauce or other food preparation that is added to food to impart a particular flavor, to enhance its flavor,[1] or in some cultures, to complement the dish. The term originally described pickled or preserved foods, but has shifted meaning over time.[2]



Как видим, при построении отчета LimeReport заполняет объекты данными, растягивает объекты со включенной опцией "autoHeigh" и потом подбирает высоту бэнда таким образом, чтобы уместить все объекты. Если опция "autoHeigh" у бэнда отключена, то подбор высоты бэнда не производится, и бэнд выводится с той высотой, что была установлена в дизайнере.

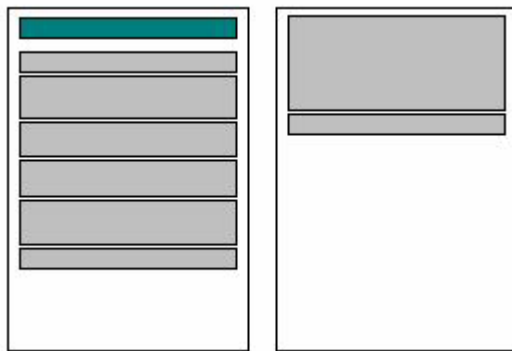
Если мы попробуем отключить эту опцию, мы увидим, что объекты с длинным текстом по-прежнему растягиваются, а бэнды - нет, что приводит также к обрезке текста.



## 7.9 Разрыв данных

Обратим внимание на одну особенность отчета с категориями: на некоторых страницах внизу остается много пустого места. Почему это происходит? Когда отчет строится, ядро LimeReport заполняет свободное место листа бэндами. После вывода каждого бэнда текущая позиция смещается все ниже и ниже. Когда LimeReport обнаруживает, что места для вывода очередного бэнда не хватает (его высота больше, чем высота оставшегося места на листе), то формируется новая страница и вывод бэндов продолжается на ней. И так до тех пор, пока есть записи в наборе данных

Наш отчет как раз содержит объект с большим количеством текста, поэтому высота бэндов получается довольно большая. И если большой бэнд не помещается на страницу, он переносится на следующую, а внизу страницы остается много неиспользованного места. Это видно на следующем рисунке:



Чтобы рациональнее использовать бумагу, воспользуемся возможностью LimeReport разбивать содержимое бэндов на части. Все, что нужно - это включить опцию "splittable" у бэнда "Data". Мы видим, что пустого места внизу страниц отчета значительно поубавилось:











Следует отметить, что алгоритм разрыва не обеспечивает 100% качества получаемого отчета. Поэтому используйте эту опцию аккуратно.

### 7.10 Печать данных в виде таблицы

Часто бывает необходимо отобразить отчет в виде таблицы с оформлением. Один из примеров такого отчета – это прайс-лист. Чтобы построить такой отчет в LimeReport, надо всего лишь включить оформление у объектов, лежащих на бэнде "Данные". Рассмотрим несколько вариантов оформления на примере тестового отчета. Создадим отчет следующего вида:



Разместим объекты на бэнде встык, а также уменьшим высоту бэнда до минимального размера. Первый и самый простой тип таблицы – с полным оформлением. Для этого надо у каждого объекта включить все линии рамки:

1	Beverages	
2	Condiments	
3	Confections	
4	Dairy Products	
5	Grains/Cereals	
6	Meat/Poultry	
7	Produce	
8	Seafood	

Следующий тип оформления – только горизонтальные или только вертикальные линии – делается аналогично, у объектов включается горизонтальное или вертикальное оформление.

1	Beverages	
2	Condiments	
3	Confections	
4	Dairy Products	
5	Grains/Cereals	
6	Meat/Poultry	
7	Produce	
8	Seafood	

Все вышеприведенные примеры содержали бэнды, которые имели фиксированный размер. Но как вывести таблицу, если бэнд растягиваемый? Покажем это на примере. Включим свойство `autoSize` у объекта "Изображение". В этом случае высота бэнда будет подбираться в зависимости от размера изображения. Мы получим отчет следующего вида:

1	Beverages	
2	Condiments	
3	Confections	
4	Dairy Products	

Немного не то, что нам нужно – хотелось бы, чтобы рамки соседних объектов тоже растягивались. LimeReport позволяет легко решить эту проблему. Для построения подобных отчетов достаточно включить у всех объектов, которые должны быть растянуты, свойство `"stretchToMaxHeight"`. При этом ядро LimeReport сначала считает максимальную высоту бэнда, затем "дотягивает" объекты с включенной опцией до нижнего края бэнда. Т.к. вместе с объектом растягивается и его рамка, в результате вид отчета меняется:

1	Beverages	
2	Condiments	
3	Confections	
4	Dairy Products	

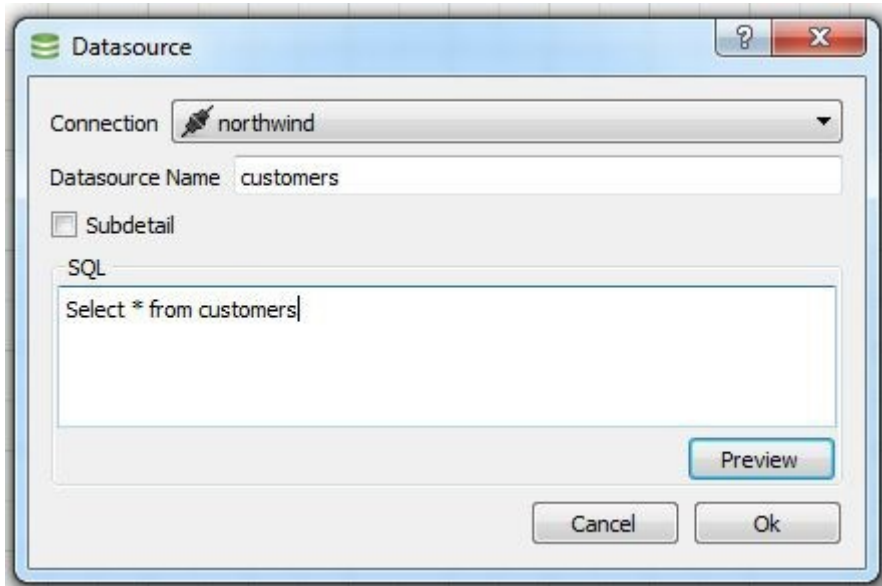
## 7.11 Отчет с двумя уровнями данных (master-detail)

До сих пор мы рассматривали отчеты, в которых присутствовал только один дата-бэнд – "Data". Это давало возможность печатать данные из одной таблицы БД. LimeReport позволяет печатать отчеты, содержащие неограниченное количество уровней данных. Рассмотрим создание двухуровневого отчета. Он будет содержать данные из таблиц Customer и Orders.

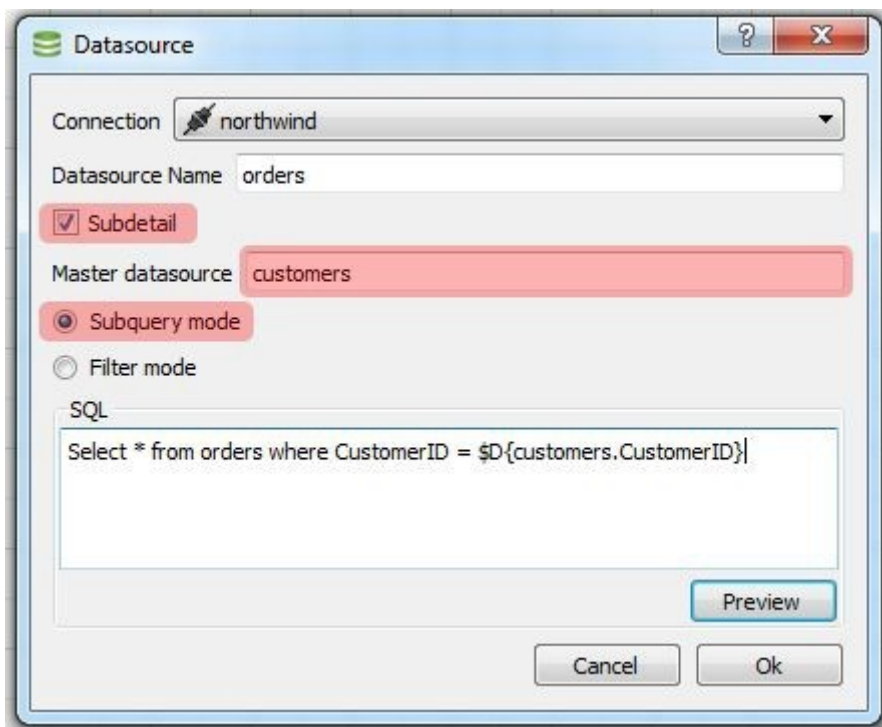
Первая таблица – это список клиентов, вторая – список заказов, сделанных клиентами. Вторая таблица содержит список всех заказов, сделанных всеми компаниями. Чтобы получить список заказов, сделанных конкретной компанией, из таблицы следует отобрать записи, у которых поле CustomerID = идентификатору выбранной компании. Отчет, построенный на таких данных, будет выглядеть следующим образом:

ALFKI		Alfreds Futterkiste
ALFKI	10062	2005-04-23
ALFKI	10643	2005-04-16
ALFKI	10692	2005-03-31
ALFKI	10702	2005-02-19
ALFKI	10835	2005-03-03
ALFKI	10952	2005-02-16
ALFKI	11011	2005-04-20
ANATR		Ana Trujillo Emparedados y helados
ANATR	10308	2005-04-04
ANATR	10625	2005-04-03
ANATR	10759	2005-02-04
ANATR	10926	2005-05-01

Приступим к созданию отчета. Первым делом создадим соединение с базой данных “northwind” (Аналогично тому как было показано ранее). Далее создадим набор данных “customers”.




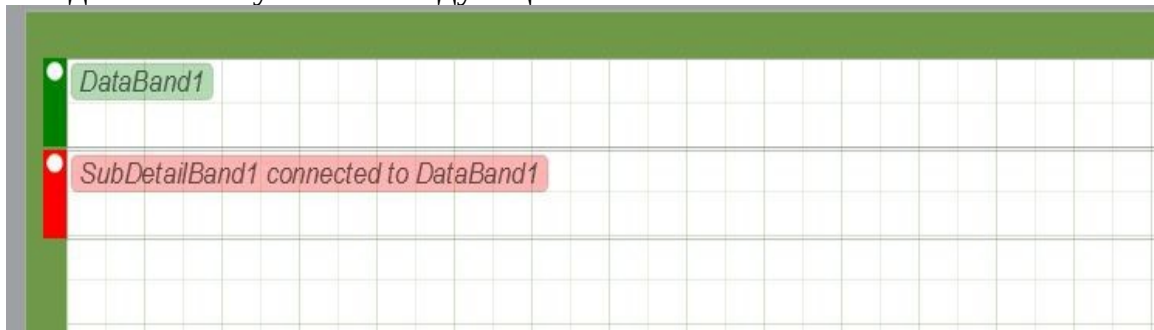
Далее создадим связанный с “customers” набор данных “orders”. Обратите ваше внимание на что параметр “Subdetail” отмечен галочкой, в качестве главного набора данных, указан набор данных “customers”, а также выбран режим “Subquery mode”



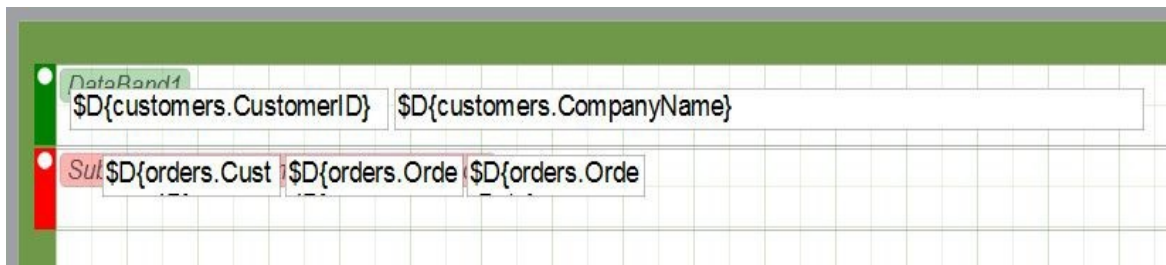
После того, как все наборы данных подключены к отчету, добавляем бэнд “Data” и подключаем её к набору данных “customers” ( параметр “datasource” устанавливается с помощью обозревателя объектов). Затем нам необходимо добавить “SubDetail” бэнд.

Нужно отметить что этот бэнд возможно вставить только в том случае если бэнд к которому “SubDetail” будет привязан активен (Левый клик мыши по “Data” бэнду).

Нажимаем вставить бэнд  и выбираем “SubDetail”. Таким образом у нас должен получиться следующий отчет.



Размещаем на “DataBand1” поля “CustomerID”, “CompanyName” из набора данных “customer”. Далее размещаем поля “CustomerID”, “OrderID”, “OrderDate” из набора данных “orders”. Отчет готов.



Запускаем предварительный просмотр и получаем:

CustomerID	CompanyName	OrderID	OrderDate
ALFKI	Alfreds Futterkiste		
ALFKI		10062	2005-04-23
ALFKI		10643	2005-04-16
ALFKI		10692	2005-03-31
ALFKI		10702	2005-02-19
ALFKI		10835	2005-03-03
ALFKI		10952	2005-02-16
ALFKI		11011	2005-04-20
ANATR	Ana Trujillo Emparedados y helados		
ANATR		10308	2005-04-04
ANATR		10625	2005-04-03
ANATR		10759	2005-02-04
ANATR		10926	2005-05-01

Продолжение следует ....